51CTO.com
技术博客Blog

博客月刊

blog.51cto.com

2014年9月
总第 07 期

- 部署Lync Server 2013 SQL见证服务器
- 大数据时代的全能日志分析专家--Splunk安装与实践
- 一次支付平台紧急故障处理备忘
- Oracle 11g rac 生产环境部署详录
- 如何才能让技术变得更有价值？

目录

安全运维之：Linux 后门入侵检测工具，附最新 bash 漏洞解决方

作者：高俊峰 来源：http://ixdba.blog.51cto.com/2895551/1557983

一、rootkit 简介

rootkit 是 Linux 平台下最常见的一种木马后门工具，它主要通过替换系统文件来达到入侵和和隐蔽的目的，这种木马比普通木马后门更加危险和隐蔽，普通的检测工具和检查手段很难发现这种木马。rootkit 攻击能力极强，对系统的危害很大，它通过一套工具来建立后门和隐藏行迹，从而让攻击者保住权限，以使它在任何时候都可以使用 root 权限登录到系统。

rootkit 主要有两种类型：文件级别和内核级别，下面分别进行简单介绍。

1、文件级别 rootkit

文件级别的 rootkit 一般是通过程序漏洞或者系统漏洞进入系统后，通过修改系统的重要文件来达到隐藏自己的目的。在系统遭受 rootkit 攻击后，合法的文件被木马程序替代，变成了外壳程序，而其内部是隐藏着的后门程序。通常容易被 rootkit 替换的系统程序有 login、ls、ps、ifconfig、du、find、netstat 等，其中 login 程序是最经常被替换的，因为当访问 Linux 时，无论是通过本地登录还是远程登录，/bin/login 程序都会运行，系统将通过/bin/login 来收集并核对用户的账号和密码，而 rootkit 就是利用这个程序的特点，使用一个带有根权限后门密码的/bin/login 来替换系统的/bin/login，这样攻击者通过输入设定好的密码就能轻松进入系统。此时，即使系统管理员修改 root 密码或者清除 root 密码，攻击者还是一样能通过 root 用户登录系统。攻击者通常在进入 Linux 系统后，会进行一系列的攻击动作，最常见的是安装嗅探器收集本机或者网络中其他服务器的重要数据。在默认情况下，Linux 中也有一些系统文件会监控这些工具动作，例如 ifconfig 命令，所以，攻击者为了避免被发现，会想方设法替换其他系统文件，常见的就是 ls、ps、ifconfig、du、find、netstat 等。如果这些文件都被替换，那么在系统层面就很难发现 rootkit 已经在系统中运行了。

这就是文件级别的 rootkit，对系统维护很大，目前最有效的防御方法是定期对系统重要文件的完整性进行检查，如果发现文件被修改或者被替换，那么很可能系统已经遭受了 rootkit 入侵。检查件完整性的工具很多，常见的有 Tripwire、 aide 等，可以通过这些工具定期检查文件系统的完整性，以检测系统是否被 rootkit 入侵。

2、内核级别的 rootkit

内核级 rootkit 是比文件级 rootkit 更高级的一种入侵方式，它可以使攻击者获得对系统底层的完全控制权，此时攻击者可以修改系统内核，进而截获运行程序向内核提交的命令，并将其重定向到入侵者所选择的程序并运行此程序，也就是说，当用户要运行程序 A 时，被入侵者修改过的内核会假装执行 A 程序，而实际上却执行了程序 B。

内核级 rootkit 主要依附在内核上，它并不对系统文件做任何修改，因此一般的检测工具很难检测到它的存在，这样一旦系统内核被植入 rootkit，攻击者就可以对系统为所欲为而不被发现。目前对于内核级的 rootkit 还没有很好的防御工具，因此，做好系统安全防范就非常重要，将系统维持在最小权限内工作，只要攻击者不能获取 root 权限，就无法在内核中植入 rootkit。

二、rootkit 后门检测工具 chkrootkit

chkrootkit 是一个 Linux 系统下查找并检测 rootkit 后门的工具，它的官方址:

http://www.chkrootkit.org/。 chkrootkit 没有包含在官方的 CentOS 源中，因此要采取手动编译的方法来安装，不过这种安装方法也更加安全。下面简单介绍下 chkrootkit 的安装过程。

1.准备 gcc 编译环境

对于 CentOS 系统，需要安装 gcc 编译环境，执行下述三条命令：

```
[root@server ~]# yum -y install gcc
[root@server ~]# yum -y install gcc-c++
[root@server ~]# yum -y install make
```

2、安装 chkrootkit

为了安全起见，建议直接从官方网站下载 chkrootkit 源码，然后进行安装，操作如下：

```
[root@server ~]# tar zxvf chkrootkit.tar.gz
[root@server ~]# cd chkrootkit-*
[root@server chkrootkit-0.50]# make sense
```

注意，上面的编译命令为 make sense

```
[root@server chkrootkit-0.50]# cd ..
[root@server ~]# cp -r chkrootkit-* /usr/local/chkrootkit
[root@server ~]# rm -rf chkrootkit-*
```

3、使用 chkrootkit

安装完的 chkrootkit 程序位于/usr/local/chkrootkit 目录下，执行如下命令即可显示 chkrootkit 的详细用法：

[root@server chkrootkit]# /usr/local/chkrootkit/chkrootkit -h

chkrootkit 各个参数的含义如下所示。

参数含义

-h 显示帮助信息

-v 显示版本信息

-l 显示测试内容

-ddebug 模式，显示检测过程的相关指令程序

-q 安静模式，只显示有问题的内容

-x 高级模式，显示所有检测结果

-r dir 设置指定的目录为根目录

-p dir1:dir2:dirN 指定 chkrootkit 检测时使用系统命令的目录

-n 跳过 NFS 连接的目录

chkrootkit 的使用比较简单，直接执行 chkrootkit 命令即可自动开始检测系统。下面是某个系统的检测结果：

```
[root@server chkrootkit]# /usr/local/chkrootkit/chkrootkit
 Checking `ifconfig'... INFECTED
Checking `ls'... INFECTED
Checking `login'... INFECTED
Checking `netstat'... INFECTED
Checking `ps'... INFECTED
Checking `top'... INFECTED
Checking `sshd'... not infected
Checking `syslogd'... not tested
Checking `tar'... not infected
Checking `tcpd'... not infected
Checking `tcpdump'... not infected
Checking `telnetd'... not found
```

从输出可以看出，此系统的 ifconfig、ls、login、netstat、ps 和 top 命令已经被感染。针对被感染 rootkit 的系统，最安全而有效的方法就是备份数据重新安装系统。

4、chkrootkit 的缺点

chkrootkit 在检查 rootkit 的过程中使用了部分系统命令，因此，如果服务器被黑客入侵，那么依赖的系统命令可能也已经被入侵者替换，此时 chkrootkit 的检测结果将变得完全不可信。为了避免 chkrootkit 的这个问题，可以在服务器对外开放前，事先将 chkrootkit 使用的系统命令进行备份，在需要的时候使用备份的原始系统命令让 chkrootkit 对 rootkit 进行检测。这个过程可以通过下面的操作实现：

```
[root@server ~]# mkdir /usr/share/.commands
[root@server ~]# cp `which --skip-alias awk cut echo find egrep id head ls netstat ps strings sed
uname` /usr/share/.commands
[root@server ~]# /usr/local/chkrootkit/chkrootkit -p /usr/share/.commands/
[root@server share]# cd /usr/share/
[root@server share]# tar zcvf commands.tar.gz .commands
[root@server share]#  rm -rf commands.tar.gz
```

上面这段操作是在/usr/share/下建立了一个.commands 隐藏文件，然后将 chkrootkit 使用的系统命令进行备份到这个目录下。为了安全起见，可以将.commands 目录压缩打包，然后下载到一个安全的地方进行备份，以后如果服务器遭受入侵，就可以将这个备份上传到服务器任意路径下，然后通过 chkrootkit 命令的“-p”参数指定这个路径进行检测即可。

三、rootkit 后门检测工具 RKHunter

RKHunter 是一款专业的检测系统是否感染 rootkit 的工具，它通过执行一系列的脚本来确认服务器是否已经感染 rootkit。在官方的资料中，RKHunter 可以作的事情有：

MD5 校验测试，检测文件是否有改动

检测 rootkit 使用的二进制和系统工具文件

检测特洛伊木马程序的特征码

检测常用程序的文件属性是否异常

检测系统相关的测试

检测隐藏文件

检测可疑的核心模块 LKM

检测系统已启动的监听端口

下面详细讲述下 RKHunter 的安装与使用。

1、安装 RKHunter

RKHunter 的官方网页地址为：http://www.rootkit.nl/projects/rootkit_hunter.html，建议从这个网站下载 RKHunter，这里下载的版本是 rkhunter-1.4.0.tar.gz。RKHunter 的安装非常简单，过程如下：

```
[root@server ~]# ls
rkhunter-1.4.0.tar.gz
[root@server ~]# pwd
/root
[root@server ~]# tar -zxvf rkhunter-1.4.0.tar.gz
[root@server ~]# cd rkhunter-1.4.0
[root@server rkhunter-1.4.0]# ./installer.sh  --layout default --install
```

这里采用 RKHunter 的默认安装方式，rkhunter 命令被安装到了/usr/local/bin 目录下。

2、使用 rkhunter 指令

rkhunter 命令的参数较多，但是使用非常简单，直接运行 rkhunter 即可显示此命令的用法。下面简单介绍下 rkhunter 常用的几个参数选项。

[root@server ~]#/usr/local/bin/rkhunter–help

Rkhunter 常用参数以及含义如下所示。

参数含义

-c, --check 必选参数，表示检测当前系统

--configfile <file>使用特定的配置文件

--cronjob 作为 cron 任务定期运行

--sk, --skip-keypress 自动完成所有检测，跳过键盘输入

--summary 显示检测结果的统计信息

--update 检测更新内容

-V, --version 显示版本信息

--versioncheck 检测最新版本

下面是通过 rkhunter 对某个系统的检测示例：

```
[root@server rkhunter-1.4.0]# /usr/local/bin/rkhunter   -c
[ Rootkit Hunter version 1.4.0 ]
#下面是第一部分，先进行系统命令的检查，主要是检测系统的二进制文件，因为这些文件最容易被 rootkit 攻击。显示 OK 字样表示正常，显示 Warning 表示有异常，需要引起注意，而显示“Not found”字样，一般无需理会
Checking system commands...
  Performing 'strings' command checks
Checking 'strings' command                          [ OK ]
  Performing 'shared libraries' checks
Checking for preloading variables                       [ None found ]
Checking for preloaded libraries                        [ None found ]
Checking LD_LIBRARY_PATH variable                [ Not found ]
  Performing file properties checks
Checking for prerequisites                             [ Warning ]
/usr/local/bin/rkhunter  [ OK ]
/sbin/chkconfig                                     [ OK ]
....(略)....
[Press <ENTER> to continue]
#下面是第二部分，主要检测常见的 rootkit 程序，显示“Not found”表示系统未感染此 rootkit
Checking for rootkits...
  Performing check of known rootkit files and directories
55808 Trojan - Variant A                                [ Not found ]
ADM Worm                                          [ Not found ]
AjaKit Rootkit                                        [ Not found ]
Adore Rootkit                                         [ Not found ]
aPa Kit                                             [ Not found ]
Apache Worm                                         [ Not found ]
Ambient (ark) Rootkit                                 [ Not found ]
Balaur Rootkit          [ Not found ]
BeastKit Rootkit                                      [ Not found ]
beX2 Rootkit                                          [ Not found ]
BOBKit Rootkit                  [ Not found ]
....(略)....
[Press <ENTER> to continue]
#下面是第三部分，主要是一些特殊或附加的检测，例如对 rootkit 文件或目录检测、对恶意软件检测以及对指定的内核模块检测
  Performing additional rootkit checks
Suckit Rookit additional checks                        [ OK ]
Checking for possible rootkit files and directories      [ None found ]
Checking for possible rootkit strings                   [ None found ]
```

```
  Performing malware checks
Checking running processes for suspicious files         [ None found ]
Checking for login backdoors                          [ None found ]
Checking for suspicious directories                     [ None found ]
Checking for sniffer log files                        [ None found ]
  Performing Linux specific checks
Checking loaded kernel modules                    [ OK ]
Checking kernel module names                    [ OK ]
[Press <ENTER> to continue]
#下面是第四部分，主要对网络、系统端口、系统启动文件、系统用户和组配置、SSH 配置、文件系统等进行检测
Checking the network...
  Performing checks on the network ports
Checking for backdoor ports                         [ None found ]
  Performing checks on the network interfaces
Checking for promiscuous interfaces                     [ None found ]
Checking the local host...
  Performing system boot checks
Checking for local host name                        [ Found ]
Checking for system startup files                       [ Found ]
Checking system startup files for malware                [ None found ]
  Performing group and account checks
Checking for passwd file [ Found ]
Checking for root equivalent (UID 0) accounts           [ None found ]
Checking for passwordless accounts                  [ None found ]
....(略)....
[Press <ENTER> to continue]
#下面是第五部分，主要是对应用程序版本进行检测
Checking application versions...
Checking version of GnuPG[ OK ]
Checking version of OpenSSL                       [ Warning ]
Checking version of OpenSSH                       [ OK ]
#下面是最后一部分，这个部分其实是上面输出的一个总结，通过这个总结，可以大概了解服务器目录的安全状态。
System checks summary
=====================
File properties checks...
Required commands check failed
Files checked: 137
Suspect files: 4
Rootkit checks...
Rootkits checked : 311
Possible rootkits: 0
Applications checks...
Applications checked: 3
Suspect applications: 1
The system checks took: 6 minutes and 41 seconds
```

在 Linux 终端使用 rkhunter 来检测，最大的好处在于每项的检测结果都有不同的颜色显示，如果是绿色的表示没有问题，如果是红色的，那就要引起关注了。另外，在上面执行检测的过程中，在每个部分检测完成后，需要以 Enter 键来继续。如果要让程序自动运行，可以执行如下命令：

[root@server ~]# /usr/local/bin/rkhunter --check --skip-keypress

同时，如果想让检测程序每天定时运行，那么可以在/etc/crontab 中加入如下内容：

30 09 * * * root /usr/local/bin/rkhunter --check --cronjob

这样，rkhunter 检测程序就会在每天的 9:30 分运行一次。

安全更新：

今天刚刚爆出 Bash 安全漏洞，SSH bash 紧急安全补丁！重要！

特别提醒：Linux 官方已经给出最新解决方案，已经解决被绕过的 bug，建议您尽快重新完成漏洞修补。openSUSE 镜像已经给出修复方案了。

【已确认被成功利用的软件及系统】

所有安装 GNU bash 版本小于或者等于 4.3 的 Linux 操作系统。

【漏洞描述】

该漏洞源于你调用的 bash shell 之前创建的特殊的环境变量，这些变量可以包含代码，同时会被 bash 执行。

【漏洞检测方法】

漏洞检测命令：env -i X='() { (a)=>\' bash -c 'echo date'; cat echo

修复前

输出: 当前系统时间

使用修补方案修复后

输出：

date

（备注：输出结果中见到"date"字样就修复成功了。）

特别提示：该修复不会有任何影响，如果您的脚本使用以上方式定义环境变量，修复后您的脚本执行会报错。

【建议修补方案 】

请您根据 Linux 版本选择您需要修复的命令， 为了防止意外情况发生，建议您执行命令前先对 Linux 服务器系统盘打个快照，如果万一出现升级影响您服务器使用情况，可以通过回滚系统盘快照解决。

centos:(最终解决方案)

yum clean all

yum makecache

yum -y update bash

ubuntu:(最终解决方案)

apt-get update

apt-get -y install --only-upgrade bash

debian:(最终解决方案)

7.5　64bit && 32bit

apt-get update

apt-get -y install --only-upgrade bash

6.0.x 64bit

wget http://mirrors.aliyun.com/debian/pool/main/b/bash/bash_4.1-3+deb6u2_amd64.deb &&

dpkg -i bash_4.1-3+deb6u2_amd64.deb

6.0.x 32bit

wget http://mirrors.aliyun.com/debian/pool/main/b/bash/bash_4.1-3+deb6u2_i386.deb &&

dpkg -i bash_4.1-3+deb6u2_i386.deb

aliyun linux:(最终解决方案)

5.x 64bit

wget http://mirrors.aliyun.com/centos/5/updates/x86_64/RPMS/bash-3.2-

33.el5_10.4.x86_64.rpm && rpm -Uvh bash-3.2-33.el5_10.4.x86_64.rpm

5.x 32bit

wget http://mirrors.aliyun.com/centos/5/updates/i386/RPMS/bash-3.2-33.el5_10.4.i386.rpm

&& rpm -Uvh bash-3.2-33.el5_10.4.i386.rpm

opensuse:(最终解决方案)

zypper clean

zypper refresh

zypper update -y bash

安全运维之：服务器遭受攻击后的一般处理过程

作者：高俊峰 来源：http://ixdba.blog.51cto.com/2895551/1556862

安全总是相对的，再安全的服务器也有可能遭受到攻击。作为一个安全运维人员，要把握的原则是：尽量做好系统安全防护，修复所有已知的危险行为，同时，在系统遭受攻击后能够迅速有效地处理攻击行为，最大限度地降低攻击对系统产生的影响。

一、处理服务器遭受攻击的一般思路

系统遭受攻击并不可怕，可怕的是面对攻击束手无策，下面就详细介绍下在服务器遭受攻击后的一般处理思路。

1.切断网络

所有的攻击都来自于网络，因此，在得知系统正遭受黑客的攻击后，首先要做的就是断开服务器的网络连接，这样除了能切断攻击源之外，也能保护服务器所在网络的其他主机。

2.查找攻击源

可以通过分析系统日志或登录日志文件，查看可疑信息，同时也要查看系统都打开了哪些端口，运行哪些进程，并通过这些进程分析哪些是可疑的程序。这个过程要根据经验和综合判断能力进行追查和分析。下面的章节会详细介绍这个过程的处理思路。

3.分析入侵原因和途径

既然系统遭到入侵，那么原因是多方面的，可能是系统漏洞，也可能是程序漏洞，一定要查清楚是哪个原因导致的，并且还要查清楚遭到攻击的途径，找到攻击源，因为只有知道了遭受攻击的原因和途径，才能删除攻击源同时进行漏洞的修复。

4.备份用户数据

在服务器遭受攻击后，需要立刻备份服务器上的用户数据，同时也要查看这些数据中是否隐藏着攻击源。如果攻击源在用户数据中，一定要彻底删除，然后将用户数据备份到一个安全的地方。

5.重新安装系统

永远不要认为自己能彻底清除攻击源，因为没有人能比黑客更了解攻击程序，在服务器遭到攻击后，最安全也最简单的方法就是重新安装系统，因为大部分攻击程序都会依附在系统文件或者内核中，所以重新安装系统才能彻底清除攻击源。

6.修复程序或系统漏洞

在发现系统漏洞或者应用程序漏洞后，首先要做的就是修复系统漏洞或者更改程序 bug，因为只有将程序的漏洞修复完毕才能正式在服务器上运行。

7.恢复数据和连接网络

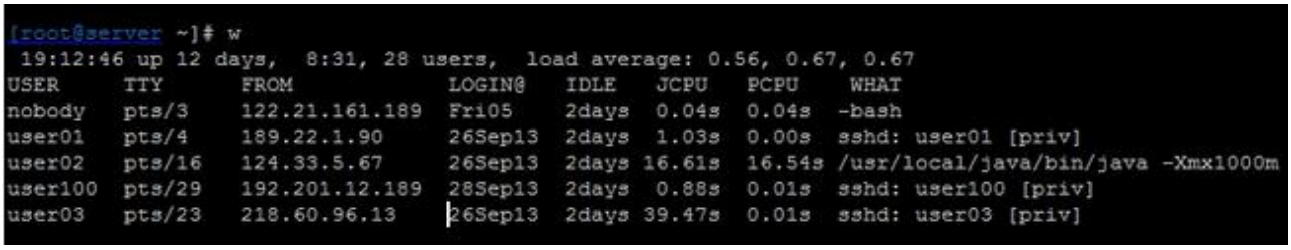
将备份的数据重新复制到新安装的服务器上，然后开启服务，最后将服务器开启网络连接，对外提供服务。

二、检查并锁定可疑用户

当发现服务器遭受攻击后，首先要切断网络连接，但是在有些情况下，比如无法马上切断网络连接时，就必须登录系统查看是否有可疑用户，如果有可疑用户登录了系统，那么需要马上将这个用户锁定，然后中断此用户的远程连接。

1.登录系统查看可疑用户

通过 root 用户登录，然后执行“w”命令即可列出所有登录过系统的用户，如下图所示。



```
[root@server ~]# w
 19:12:46 up 12 days,  8:31, 28 users,  load average: 0.56, 0.67, 0.67
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU    WHAT
nobody   pts/3    122.21.161.189   Fri05    2days  0.04s  0.04s   -bash
user01   pts/4    189.22.1.90      26Sep13  2days  1.03s  0.00s   sshd: user01 [priv]
user02   pts/16   124.33.5.67      26Sep13  2days 16.61s  16.54s /usr/local/java/bin/java -Xmx1000m
user100  pts/29   192.201.12.189   28Sep13  2days  0.88s  0.01s   sshd: user100 [priv]
user03   pts/23   218.60.96.13     26Sep13  2days 39.47s  0.01s   sshd: user03 [priv]
```

通过这个输出可以检查是否有可疑或者不熟悉的用户登录，同时还可以根据用户名以及用户登录的源地址和它们正在运行的进程来判断他们是否为非法用户。

2.锁定可疑用户

一旦发现可疑用户，就要马上将其锁定，例如上面执行“w”命令后发现 nobody 用户应该是个可疑用户（因为 nobody 默认情况下是没有登录权限的），于是首先锁定此用户，执行如下操作：

[root@server ~]# passwd -l nobody

锁定之后，有可能此用户还处于登录状态，于是还要将此用户踢下线，根据上面“w”命令的输出，即可获得此用户登录进行的 pid 值，操作如下：

[root@server ~]# ps -ef|grep @pts/3

531 6051 6049 0 19:23 ? 00:00:00 sshd: nobody@pts/3

[root@server ~]# kill -9 6051

这样就将可疑用户 nobody 从线上踢下去了。如果此用户再次试图登录它已经无法登录了。

3.通过 last 命令查看用户登录事件

last 命令记录着所有用户登录系统的日志，可以用来查找非授权用户的登录事件，而 last 命令的输出结果来源于/var/log/wtmp 文件，稍有经验的入侵者都会删掉/var/log/wtmp 以清除自己行踪，但是还是会露出蛛丝马迹在此文件中的。

三、查看系统日志

查看系统日志是查找攻击源最好的方法，可查的系统日志有/var/log/messages、/var/log/secure 等，这两个日志文件可以记录软件的运行状态以及远程用户的登录状态，还可以查看每个用户目录下的.bash_history 文件，特别是/root 目录下的.bash_history 文件，这个文件中记录着用户执行的所有历史命令。

四、**检查并关闭系统可疑进程**

检查可疑进程的命令很多，例如 ps、top 等，但是有时候只知道进程的名称无法得知路径，此时可以通过如下命令查看：

首先通过 pidof 命令可以查找正在运行的进程 PID，例如要查找 sshd 进程的 PID，执行如下命令：

```
1 [root@server ~]# pidof sshd
2 13276 12942 4284
```

然后进入内存目录，查看对应 PID 目录下 exe 文件的信息：

```
[root@server ~]# ls -al /proc/13276/exe
lrwxrwxrwx 1 root root 0 Oct  4 22:09 /proc/13276/exe -> /usr/sbin/sshd
```

这样就找到了进程对应的完整执行路径。如果还有查看文件的句柄，可以查看如下目录：

[root@server ~]# ls -al /proc/13276/fd

通过这种方式基本可以找到任何进程的完整执行信息，此外还有很多类似的命令可以帮助系统运维人员查找可疑进程。例如，可以通过指定端口或者 tcp、udp 协议找到进程 PID，进而找到相关进程：

```
[root@server ~]# fuser -n tcp 111
111/tcp:              1579
[root@server ~]# fuser -n tcp 25
25/tcp:               2037
[root@server ~]# ps -ef|grep 2037
root      2037     1  0 Sep23 ?        00:00:05 /usr/libexec/postfix/master
postfix   2046  2037  0 Sep23 ?        00:00:01 qmgr -l -t  fifo -u
postfix   9612  2037  0 20:34 ?        00:00:00 pickup -l -t fifo -u
root     14927 12944  0 21:11 pts/1    00:00:00 grep 2037
```

在有些时候，攻击者的程序隐藏很深，例如 rootkits 后门程序，在这种情况下 ps、top、netstat 等命令也可能已经被替换，如果再通过系统自身的命令去检查可疑进程就变得毫不可信，此时，就需要借助于第三方工具来检查系统可疑程序，例如前面介绍过的 chkrootkit、RKHunter 等工具，通过这些工具可以很方便的发现系统被替换或篡改的程序。

五、检查文件系统的完好性

检查文件属性是否发生变化是验证文件系统完好性最简单、最直接的方法，例如可以检查被入侵服务器上/bin/ls 文件的大小是否与正常系统上此文件的大小相同，以验证文件是否被替换，但是这种方法比较低级。此时可以借助于 Linux 下 rpm 这个工具来完成验证，操作如下：

```
[root@server ~]# rpm -Va
....L...  c /etc/pam.d/system-auth
S.5.....  c /etc/security/limits.conf
S.5....T  c /etc/sysctl.conf
S.5....T    /etc/sgml/docbook-simple.cat
S.5....T  c /etc/login.defs
S.5.....  c /etc/openldap/ldap.conf
S.5....T  c /etc/sudoers
..5....T  c /usr/lib64/security/classpath.security
....L...  c /etc/pam.d/system-auth
S.5.....  c /etc/security/limits.conf
S.5.....  c /etc/ldap.conf
S.5....T  c /etc/ssh/sshd_config
```

对于输出中每个标记的含义介绍如下：

S 表示文件长度发生了变化

M 表示文件的访问权限或文件类型发生了变化

5 表示 MD5 校验和发生了变化

D 表示设备节点的属性发生了变化

L 表示文件的符号链接发生了变化

U 表示文件/子目录/设备节点的 owner 发生了变化

G 表示文件/子目录/设备节点的 group 发生了变化

T 表示文件最后一次的修改时间发生了变化

如果在输出结果中有“M”标记出现，那么对应的文件可能已经遭到篡改或替换，此时可以通过卸载这个 rpm 包重新安装来清除受攻击的文件。

不过这个命令有个局限性，那就是只能检查通过 rpm 包方式安装的所有文件，对于通过非 rpm 包方式安装的文件就无能为力了。同时，如果 rpm 工具也遭到替换，就不能通过这个方法了，此时可以从正常的系统上复制一个 rpm 工具进行检测。

对文件系统的检查也可以通过 chkrootkit、RKHunter 这两个工具来完成，关于 chkrootkit、RKHunter 工具的使用，下次将展开介绍。

虚拟化中的 RSS 与 VMQ

作者：maomaostyle 来源：http://maomaostyle.blog.51cto.com/2220531/1547616

在之前的博文中（http://maomaostyle.blog.51cto.com/2220531/1439651 ）聊过虚拟化中的 SRIOV 技术，即单根虚拟化，在启用了 SRIOV 之后，虚拟机通过硬件实现 VF（Virtual Function）能力，使得性能近乎于物理机效果。

当然，虚拟化技术以及云计算平台已经发展的越来越快，相关的硬件加速技术也是有很多种选择，并且在不同的场合需要用户选取适合自己的解决方案，而不是盲目跟风，究竟哪些功能有用，哪些功能之间又互相冲突，都需要仔细的去评估，今天就想聊一聊有关虚拟化环境中的另外两个硬件辅助功能，RSS（接收端扩展）与 VMQ（虚拟机队列）。

##

首先在开始之前先说明一下我的演示环境：

操作系统：Windows Server 2012 R2

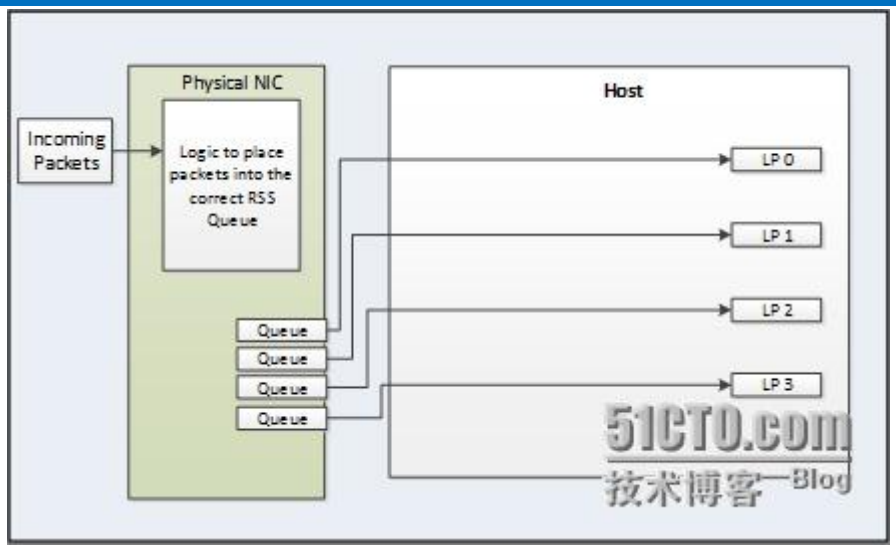
物理机：HP DL160 G6

网卡：Intel 千兆（支持 RSS 及 VMQ）

这里十分遗憾的是我的网卡是 1Gi 速率的而不是 10Gi 的，没办法屌丝只能这样了，因此无法完全说明开启 VMQ 之后的效果（具体缘由在文章后面会提到）

##

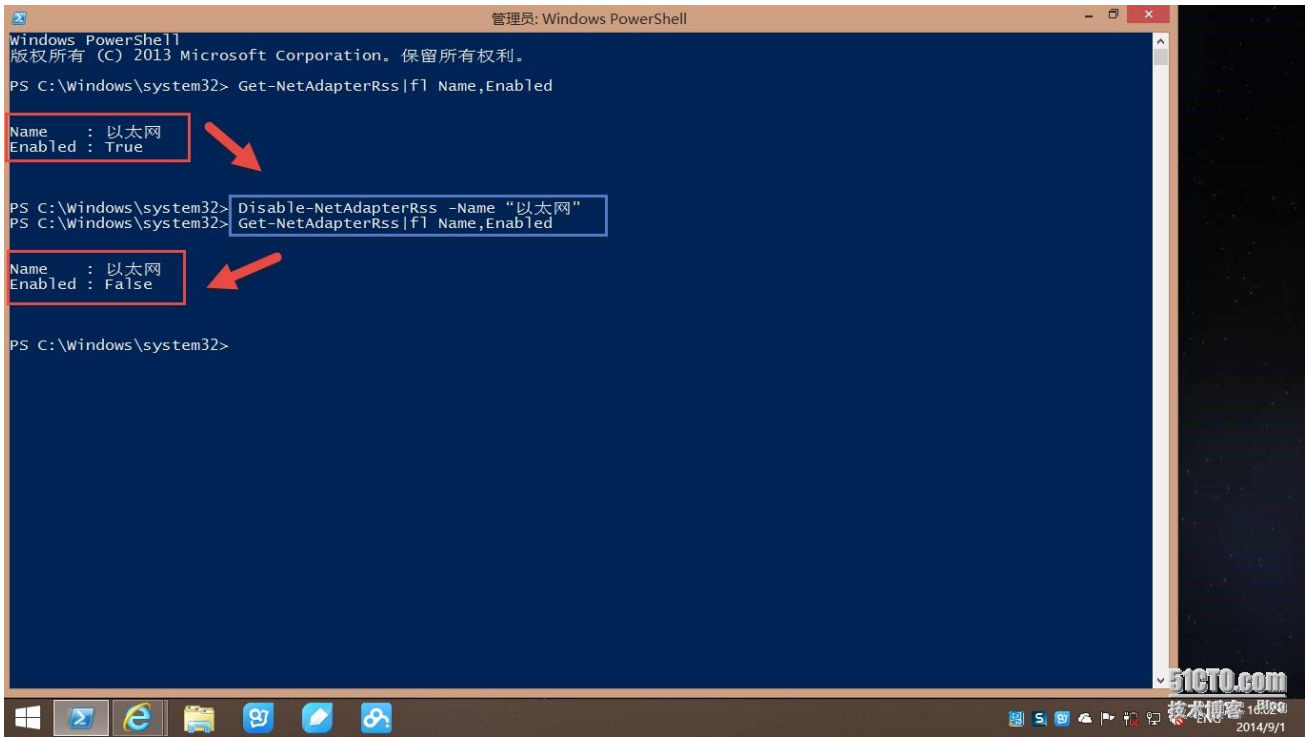
好的，现在开始进入正题，为什么要把 RSS 和 VMQ 拿到一起来说呢，因为这两个功能的初衷是比较类似的，先说 RSS，它通常是在纯物理环境下启用的，RSS 所实现的效果非常简单明了，如下图：



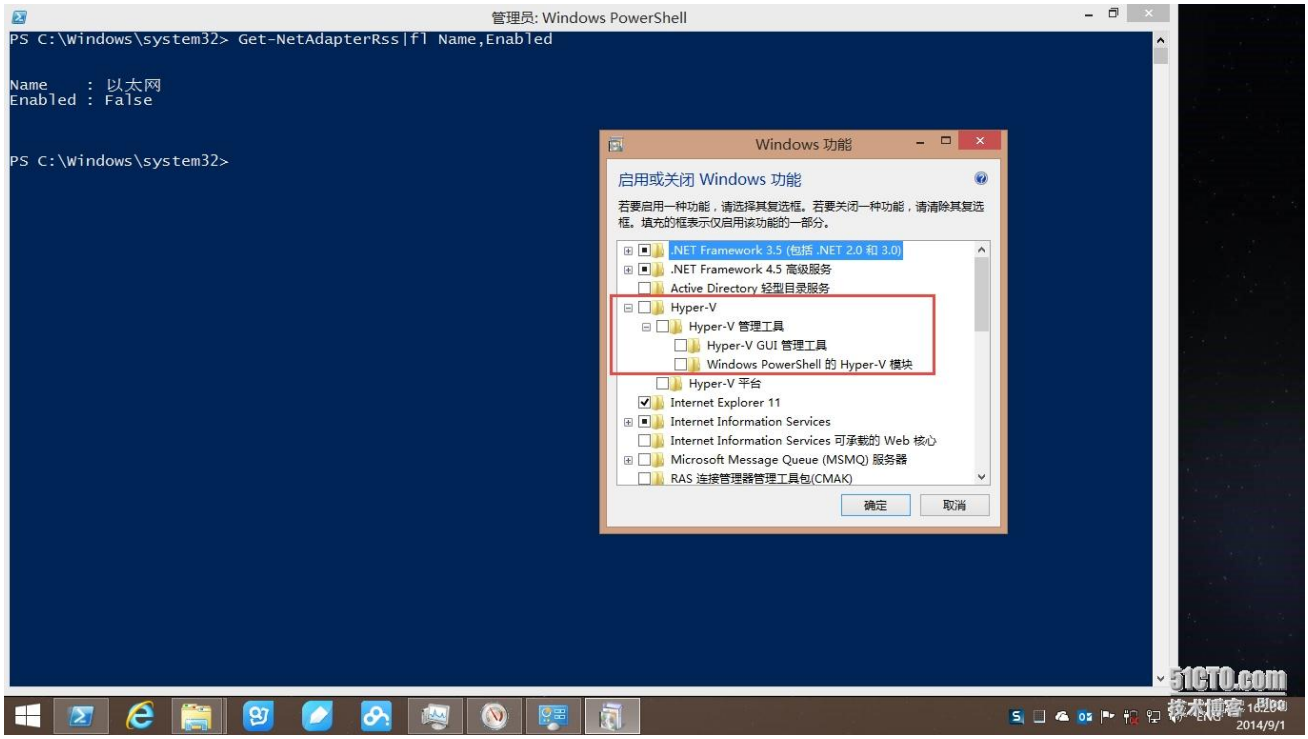
在物理 NIC 不支持 RSS 功能的时候，网络流量的负载是通过一颗逻辑 CPU 来处理的，是的你没有看错，即便你的 CPU 能力很屌，多路 N 线程也只会调用一颗 LP（logical processor）来干活儿，那么就会出现一个瓶颈问题，而 RSS 正是通过调用全部 LP 来处理网络负载来解决了这一性能问题。好就好在什么呢？如果是 1Gi 速率的话，目前这个年代单颗 LP 处理起来还是富富有余的，但众所周知，以太网发展十分迅速，10Gi 速率的出现彻底打破了这一局面，当下 OS（以 Windows 为例）调用单颗 LP 只能 run 到 3.5~4.5Gi 的能力，是的没错，也就意味着在万兆环境下，若没有 RSS 或 VMQ 的辅助，那么必将损失掉很多性能。

##

在我的 DL160 上，首先来看一下当前网卡的工作模式，RSS 属性默认都是开启的（这已经是服务器级别最基本的一个功能了），为了对比效果我先禁用它，如下图：

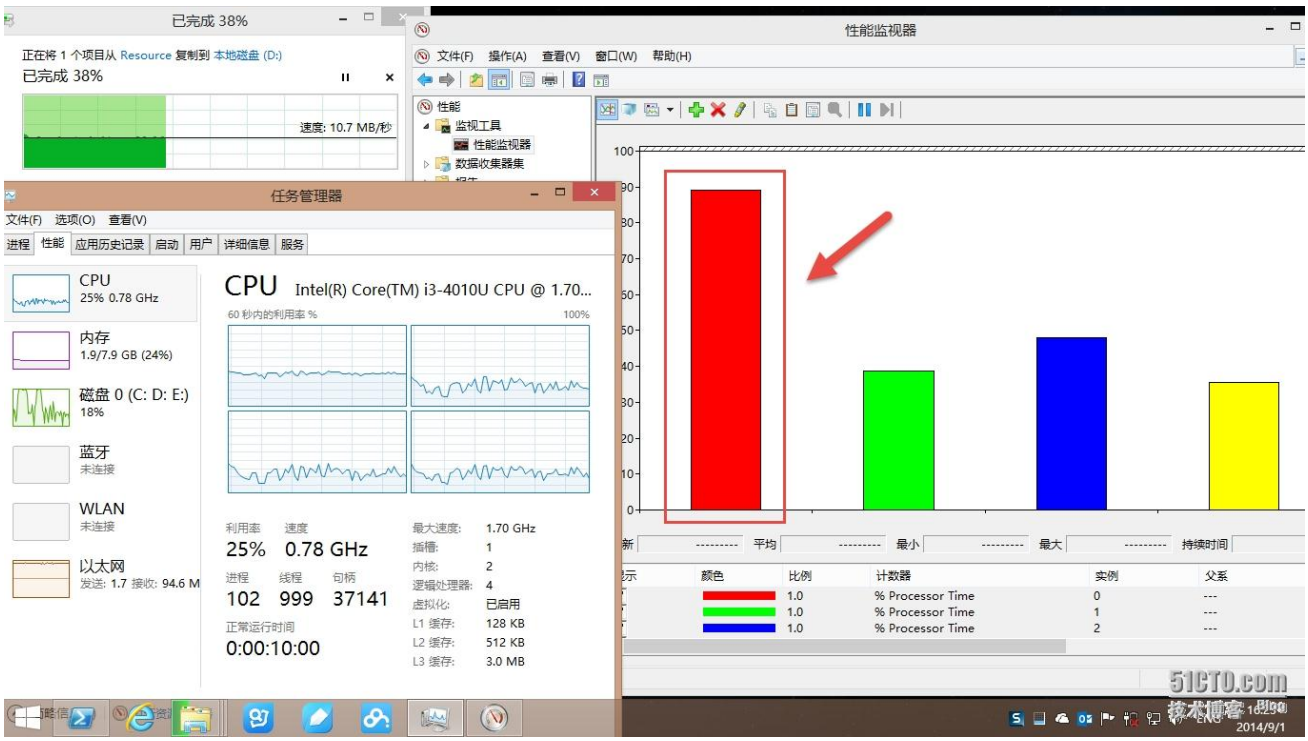


之后要确认一下我的环境尚未开启 hypervisor，也就是以 Windows 平台为例，我没有开启 Hyper-V 功能（虚拟化层），为什么要确认这一步呢，因为 RSS 在虚拟化环境中将不起作用，尽管它可能处于启用状态，但始终是无效的，为什么一会儿再说。

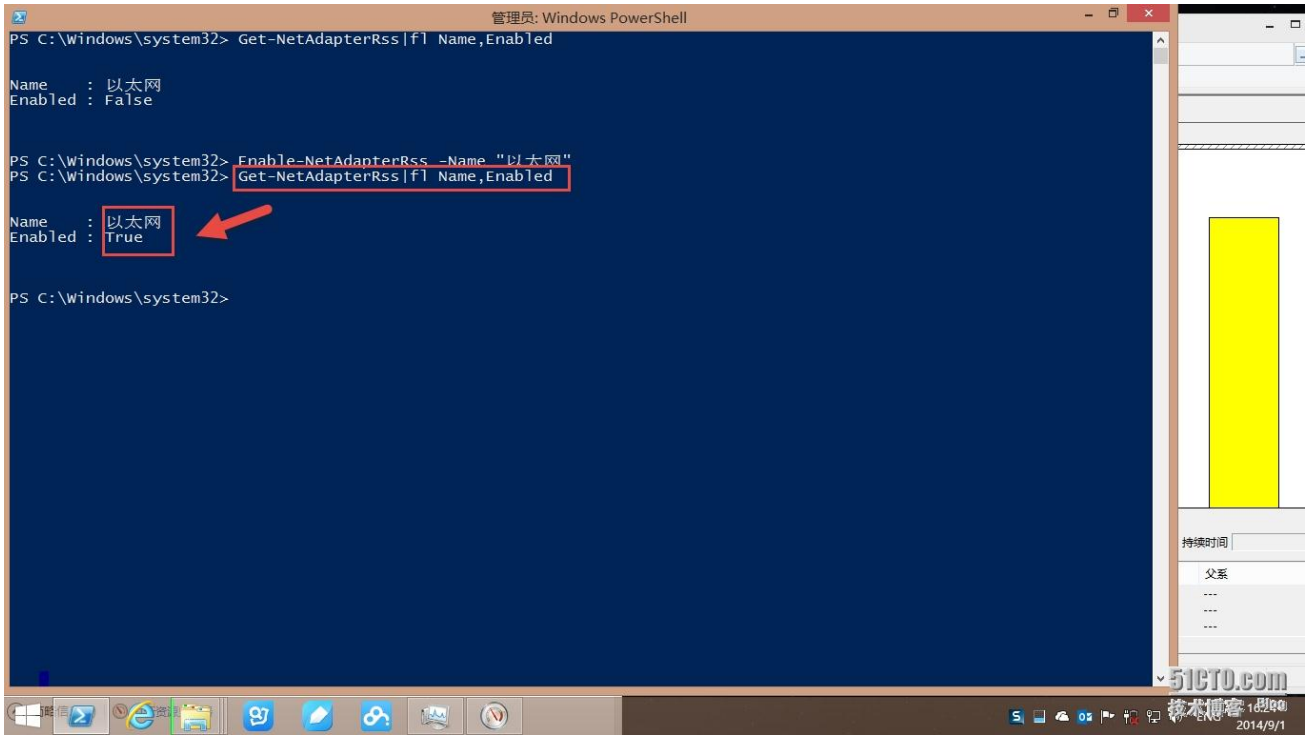


接下来我通过网络共享做一个拷贝的动作，眼尖的童鞋可能会发现我的速率是百兆的。。。是的你没看错。。。:(

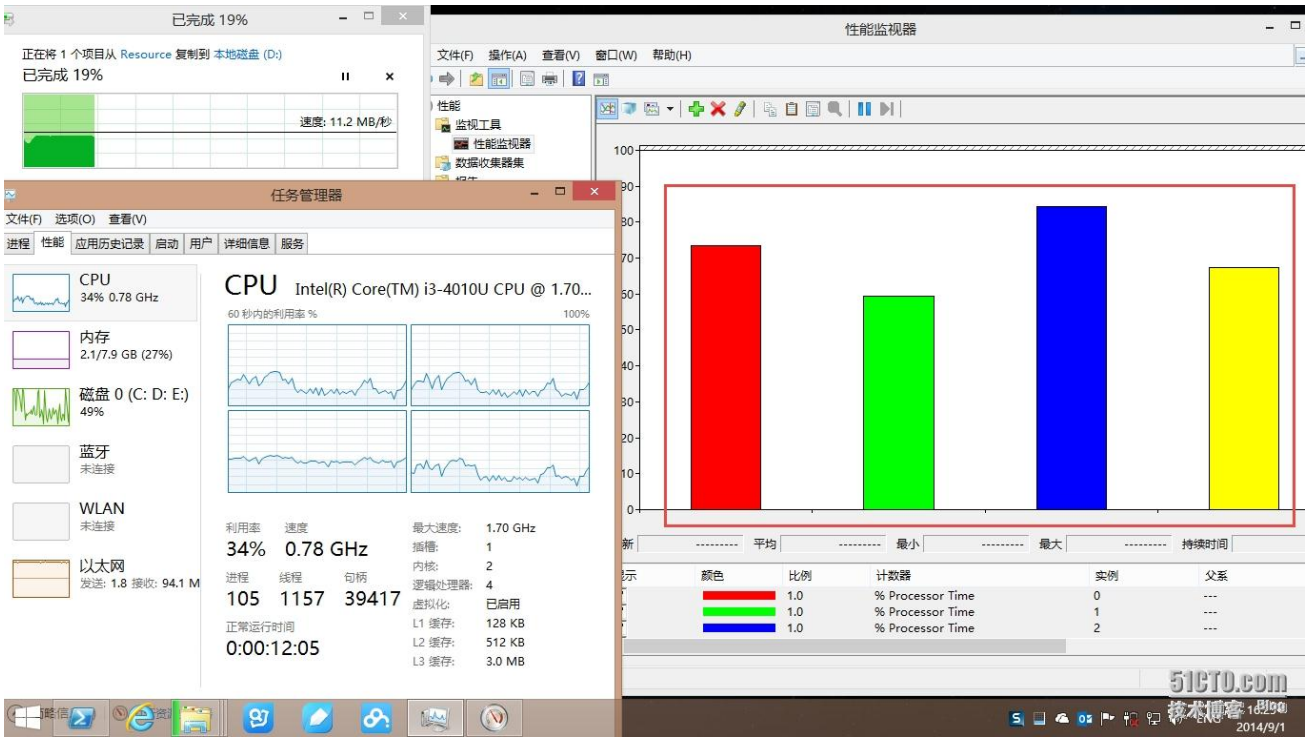
此时在 DL160 这台服务器上，大部分负载都是通过 LP0 来 run 的，通过性能计数器可以看的更明显，如下图：



我重新启用 RSS 功能做一个对比

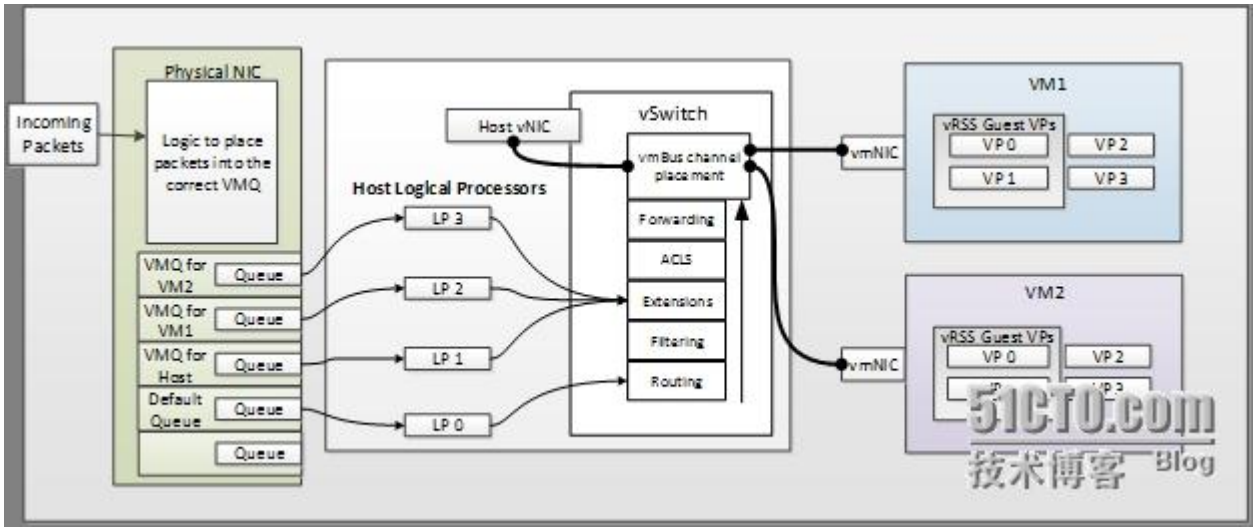


仍然是刚才的拷贝，此时服务器已经在调用所有 CPU 来处理了，我的这个截图可能看上去不是特别的明显，因为理论上来讲，RSS 功能在千兆网卡上可能也不会起作用，正如上面所讲的，单颗 LP 足以应付 1Gi 速率的负载。



##########

那么在了解了 RSS 功能之后，再来看看 VMQ，虚拟机队列顾名思义是适用于虚拟化环境下的，那为什么不能直接用 RSS 呢？看下图：



以 Windows 平台为例，在启用了 hypervisor 之后，整个基础架构发生了很大变化，首先是产生父子区域，即主机 OS 与 Guest OS，其次就是最要命的——“虚拟交换机”，virtual switch 通过路由、过滤、扩展、访问控制列表、转发、VM Bus 等堆栈组合而成，首先从系统层面的视角来看，主机 host 与 VM 之间的流量是平起平坐的，那么此时从外面进来的流量通过物理 NIC 之后需要分发到不同的目的

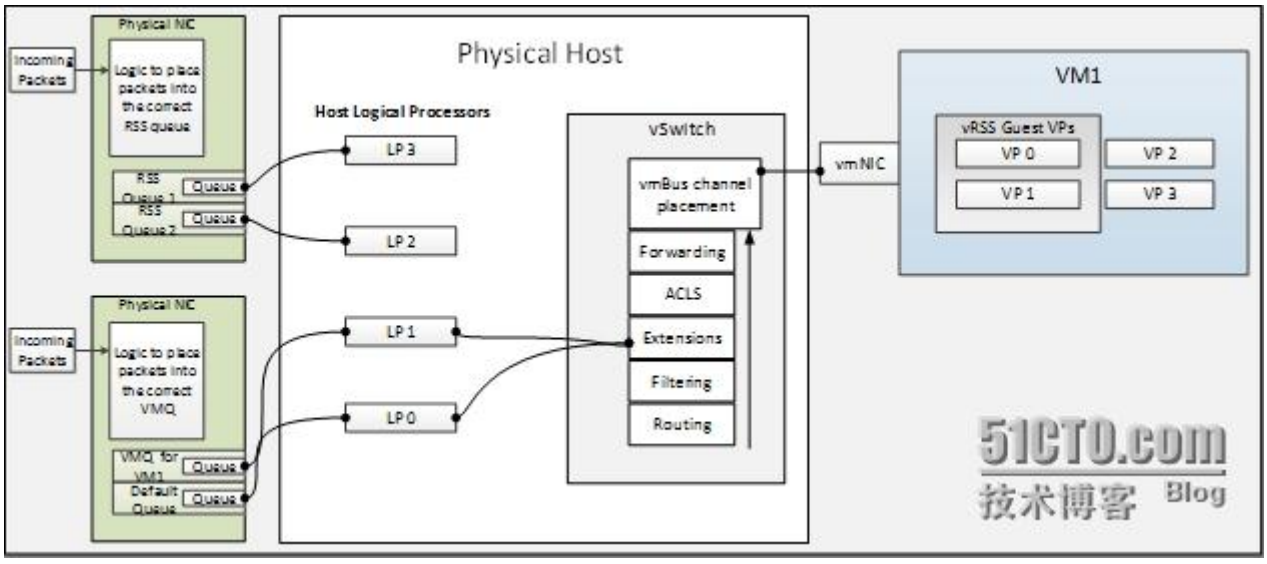
地，例如主机 OS 或者转发到某一台 VM 上，也就是说不存在针对物理机来调用所有 LP 了，那么 RSS 功能就失效了，你就算开着它也没用。

因此回到本篇的议题，为什么要把 RSS 与 VMQ 拿到一起说，因为 VMQ 正是为了解决在虚拟化环境中出现的性能瓶颈问题而诞生的，如上图所示，首先物理 NIC 需要硬件支持 VMQ 功能，然后不同厂商的不同型号，一个 NIC 口可以支持若干个队列，每一个队列将会关联到一颗 LP 上，然后这条路径将会对应到唯一的目的地，即主机 OS 或者某一台 VM。

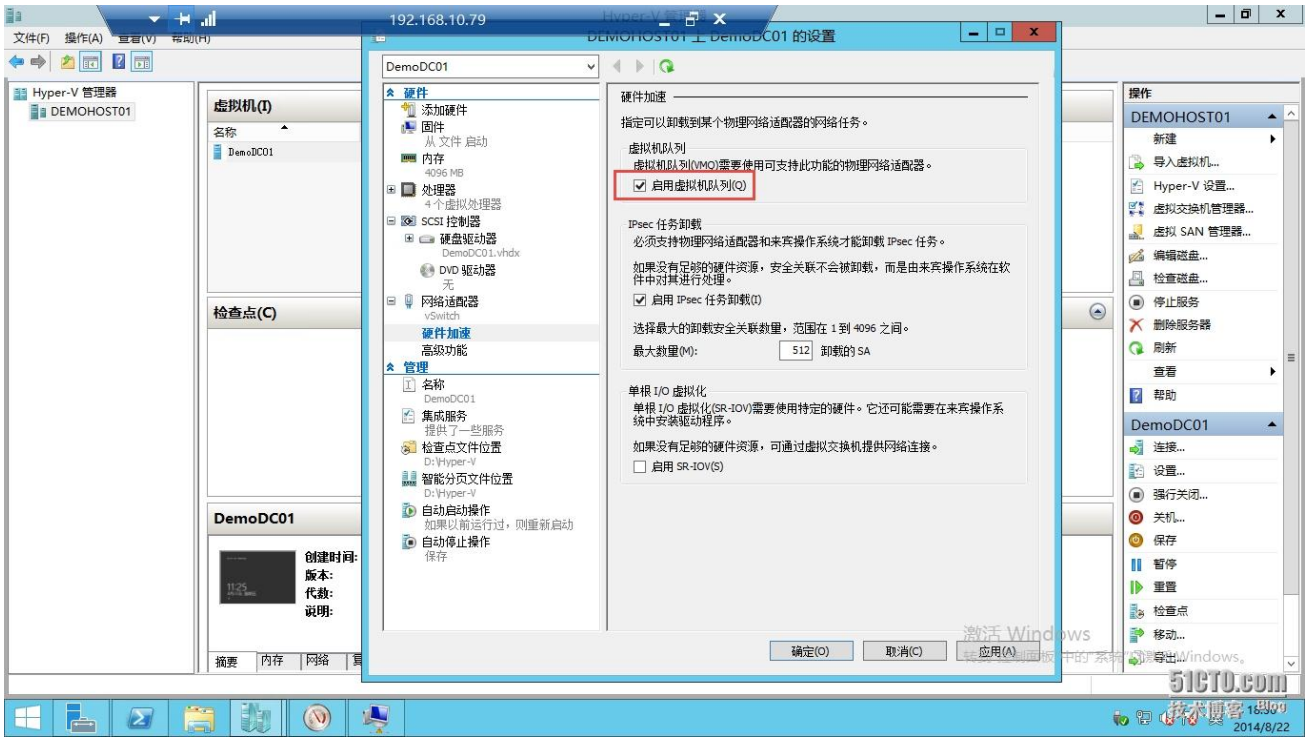
这样一来首先入方向流量的转发和路由功能将会在 NIC 上完成，不需要在虚拟交换机上做过多的停留，此外每条路径在万兆（10Gi）条件下可以发挥最高将近 5Gi 的能力，而不是所有资源共享一条 5Gi 左右链路（若不启用 VMQ 功能整个主机只由一颗 LP 负载所有 VM 和父 OS 流量）。

##

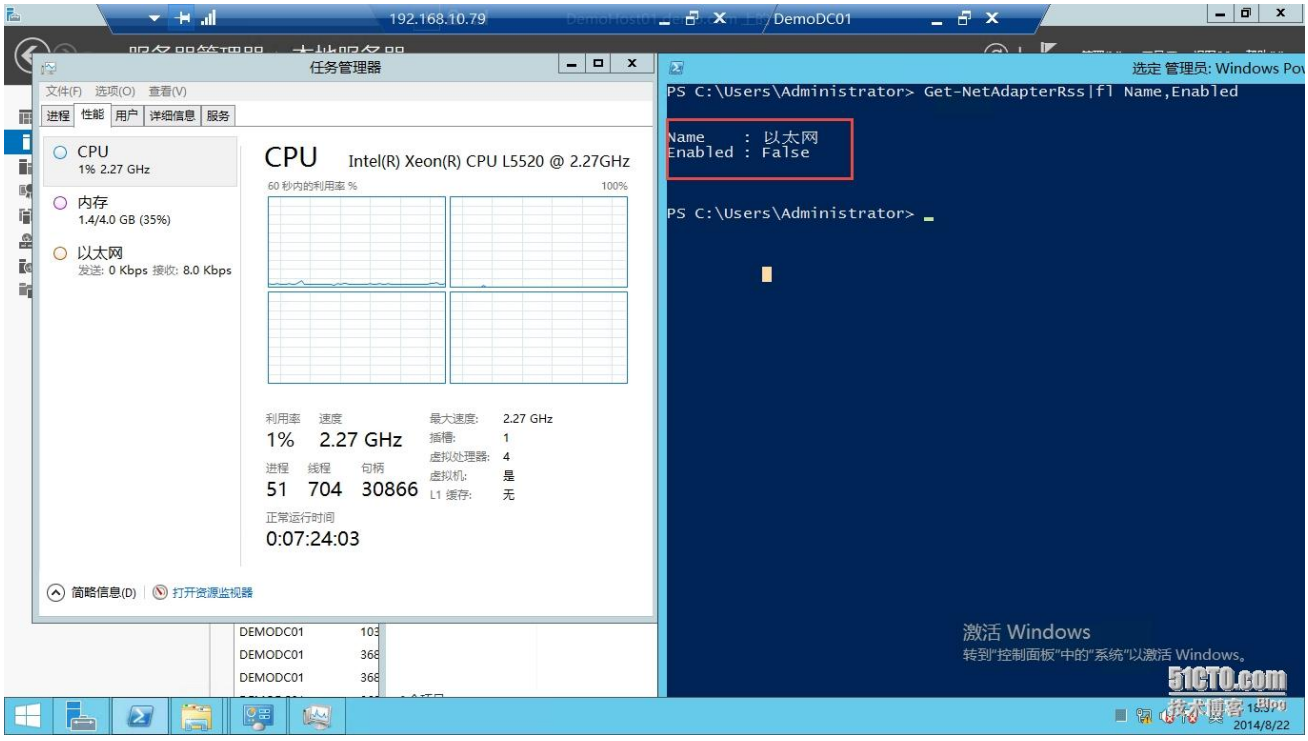
那么可能有的童鞋会想说，虽然我有了 VMQ 可以解决一部分性能问题了，但是我的 VM 可能有很多颗 VP（virtual processor），如何发挥最大性能呢？很可喜的是，在 Windows Server 2012 R2 中，Hyper-V 已经支持一项叫做 vRSS 的功能了，顾名思义，现在在虚机中我们也可以实现 RSS 的效果了。



想要实现 vRSS 的前提是，物理 NIC 要支持 VMQ 功能，并且在 Hyper-V 的设置用，确保虚拟机的 vNIC 启用了“虚拟机队列”功能。如下图：



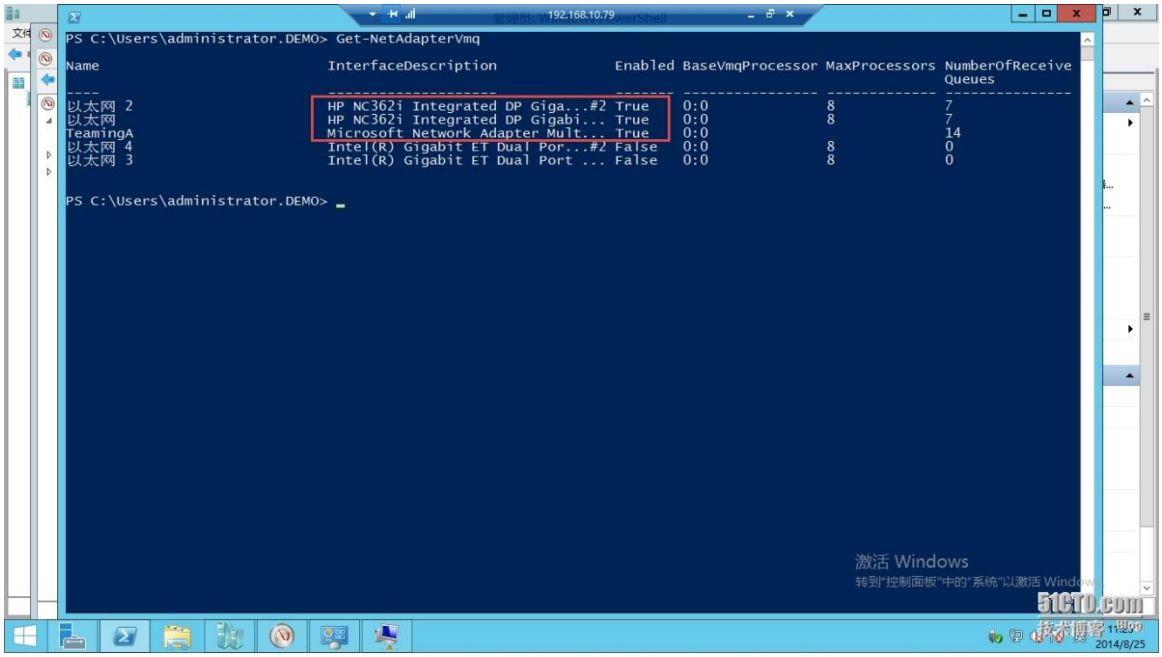
接下来看一下对比，实际想过和物理机测试 RSS 的差不多的，在 Guest OS 中查看虚拟机的 vNIC 属性，当前 vNIC 的 RSS 功能默认是关闭的。如下图：



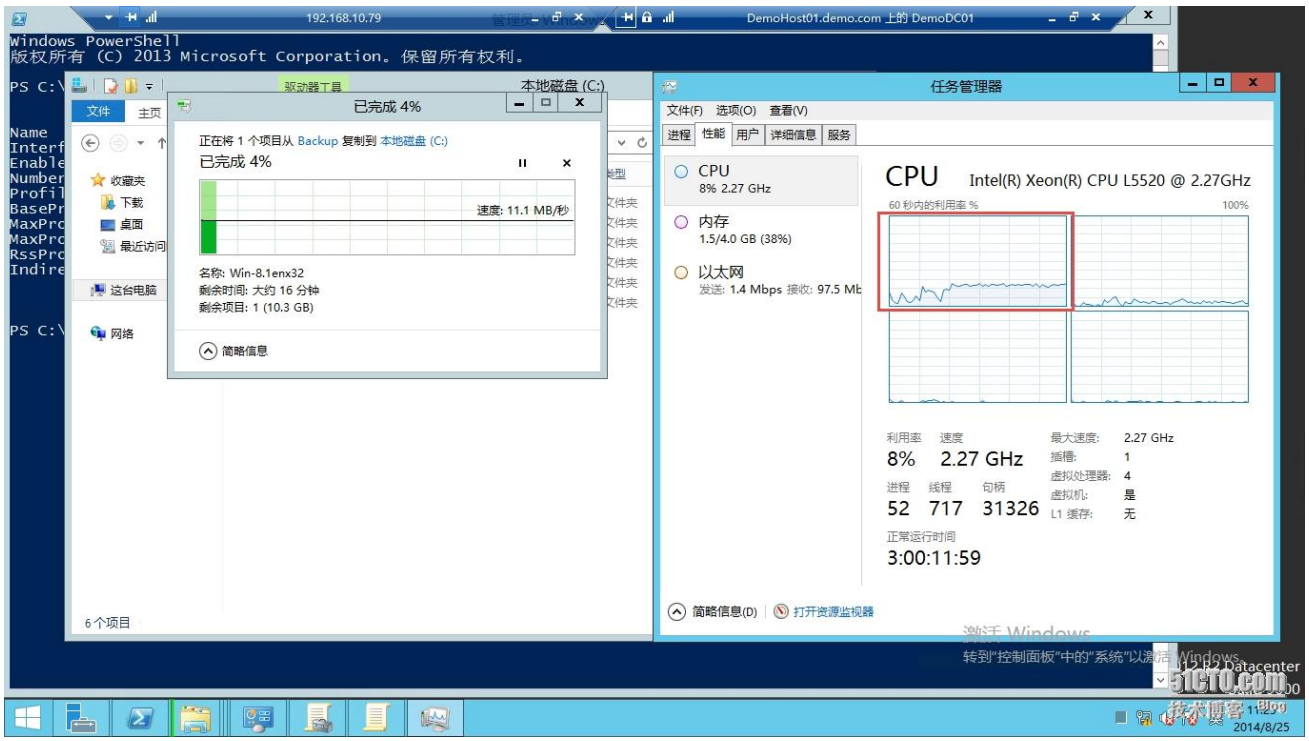
接下来查看一下宿主机上物理 NIC 的 VMQ 属性，以我的环境为例，我把“以太网”和“以太网 2”做了 Teaming，因此只需要关注两块 NIC 是否启用了 VMQ 即可。如下图：

在这里要说明一下，通过 PowerShell 返回的结果来看，我的网卡 VMQ 属性有几列字段需要重点关注一下：

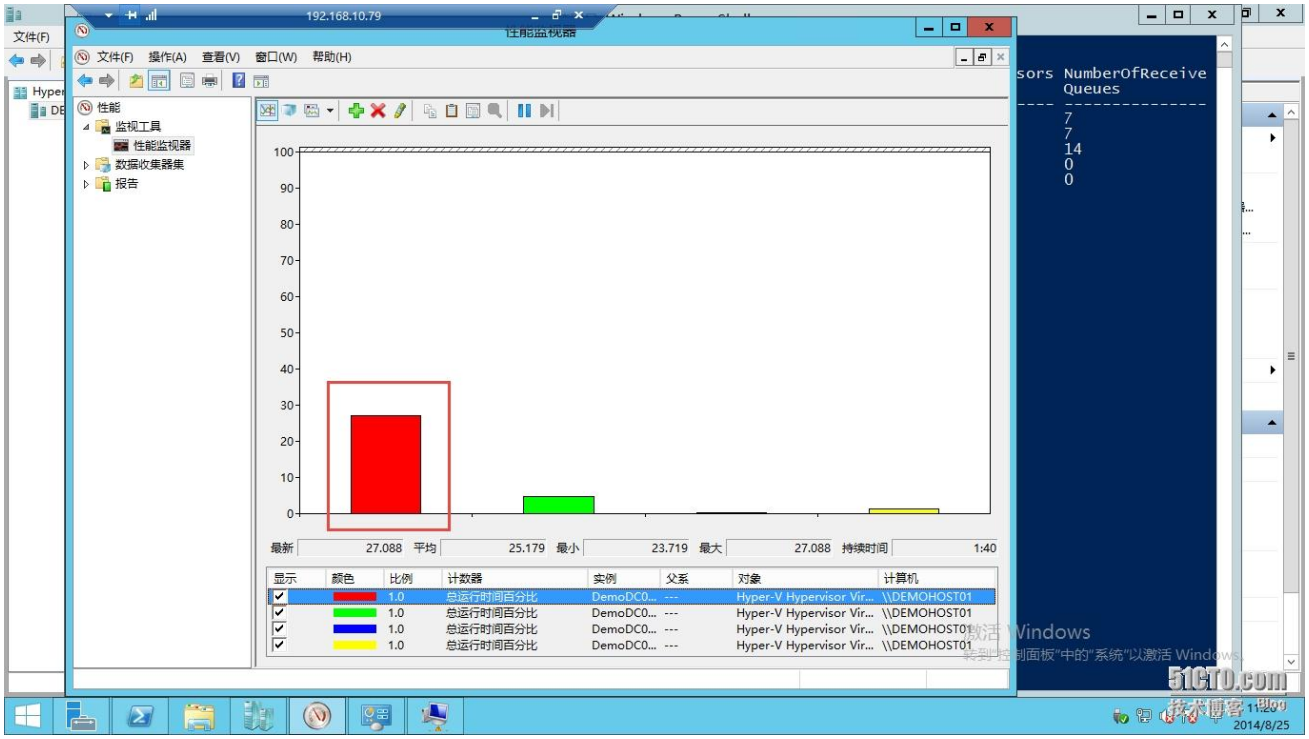
1. 首先“basevmqprocessor 0:0”是指当前“以太网”这块 NIC 会从“CPU 组 0”当中的“LP0”开始分配队列，这个组一般来讲会容纳 64 个 LP，物理机超过 64 个 LP 就会分配到下一个组。
2. 其次“maxprocessors 8”是指当前该 NIC 会调用最多 8 个 LP，那么从上一条 basevmqprocessor 0:0 来推算，这个队列会调用到 LP7,（从 LP0~LP7 总共 8 颗 LP）。
3. 最后“numberofreceivequeue 7”就是说最多支持 7 个队列，因此可以看出，并不是说 NIC 能调用 8 个 LP 就一位着它能处理 8 个队列。



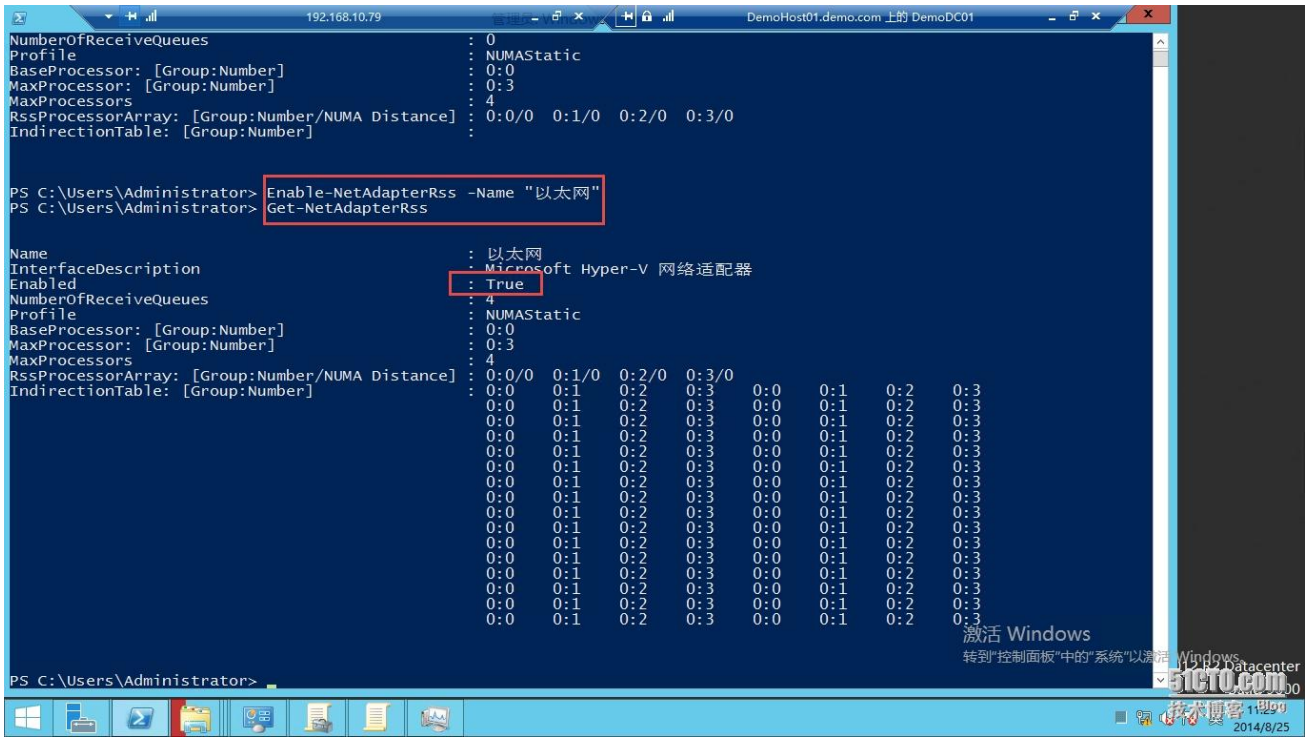
接下来在没有启用虚拟机 RSS 功能（也就是 vRSS）的前提下进行一个拷贝动作，可以再 Guest OS 看到当前只调用了 VP0。如下图：



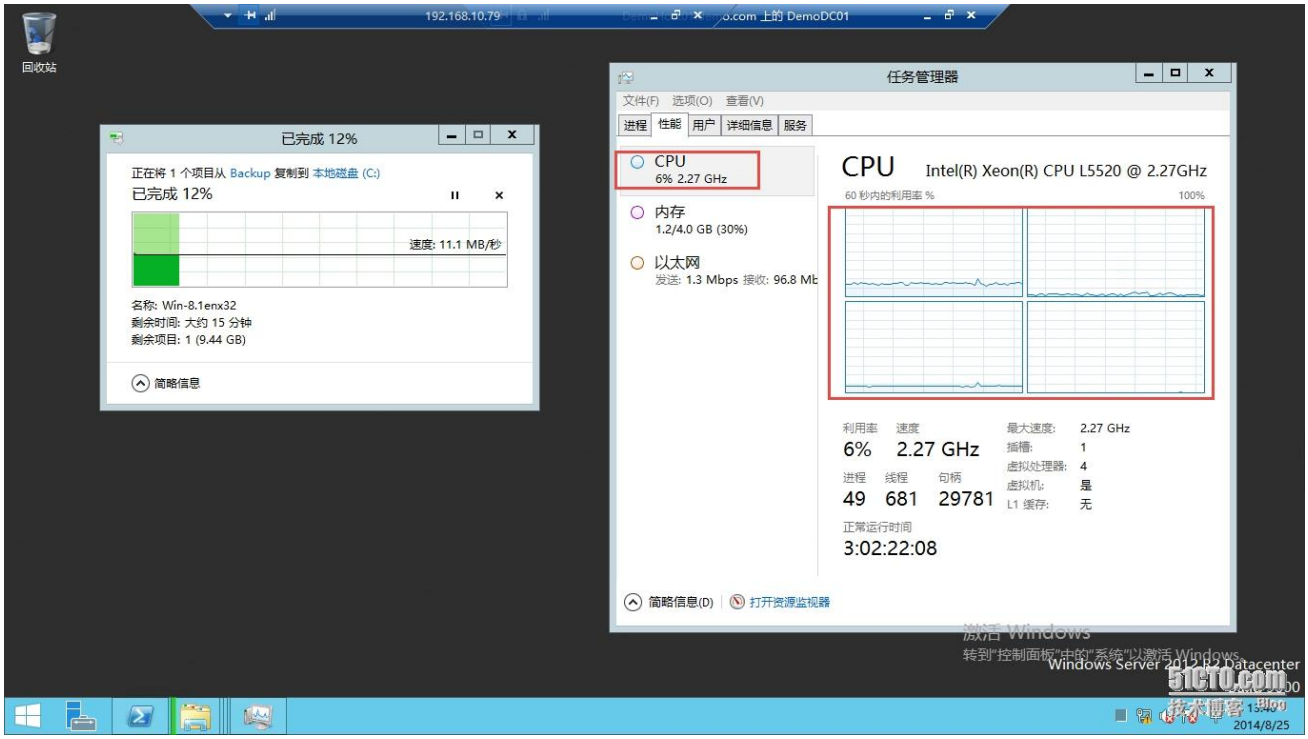
通过物理机的性能计数器来观测 VP 使用情况会更明显，如下图：



接下来启用虚拟机的 RSS 功能，即 vRSS，如下图：

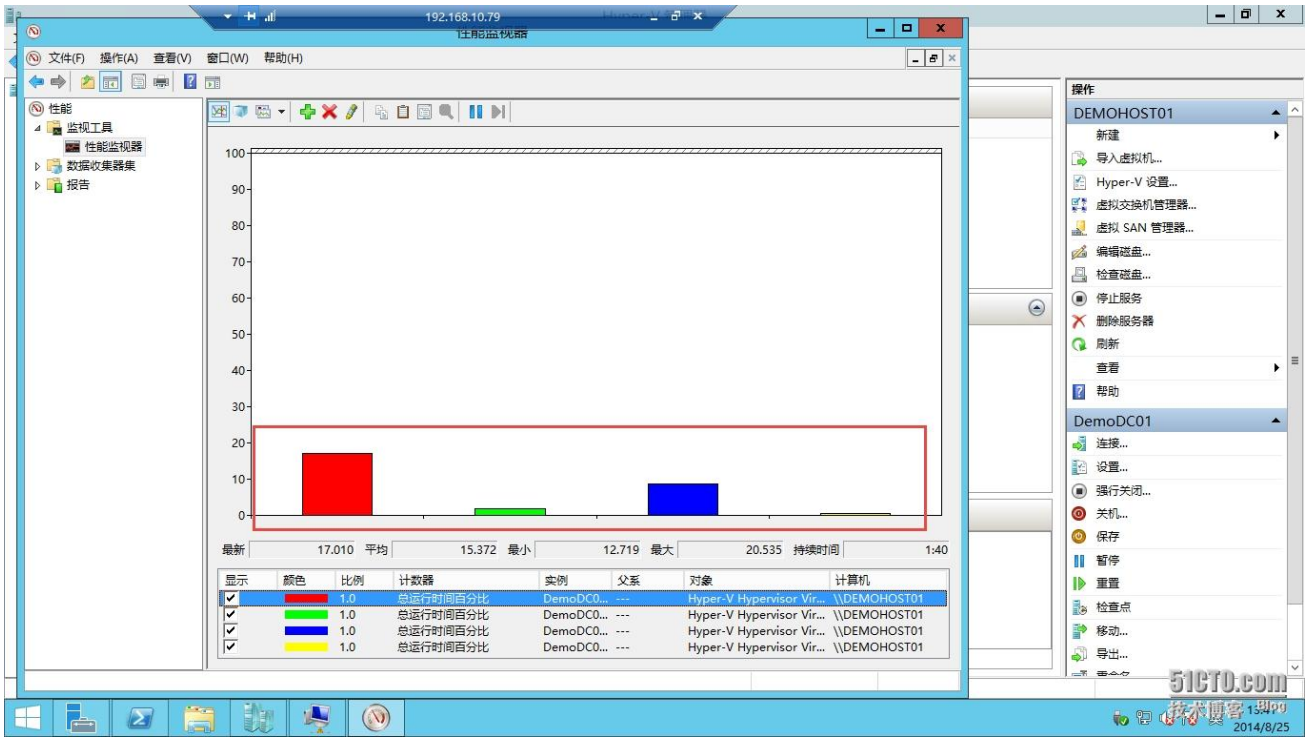


继续进行一个拷贝工作，观察 Guest OS 内 CPU 使用情况，如下图：



同样在物理机上通过性能计数器观察 VP 使用率，如下图：

若在万兆条件下测试效果会更直观



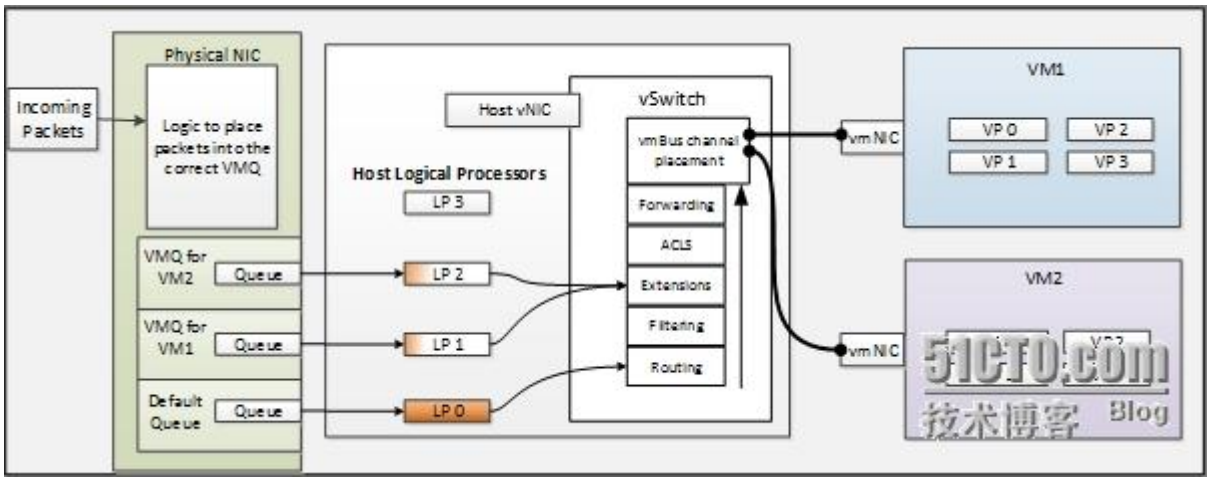
##

上面主要看了一下 RSS 与 vRSS 的一些演示，特别是 vRSS 也是要基于 VMQ 功能才可以启用的，那么下面就看一看 VMQ 的效果。

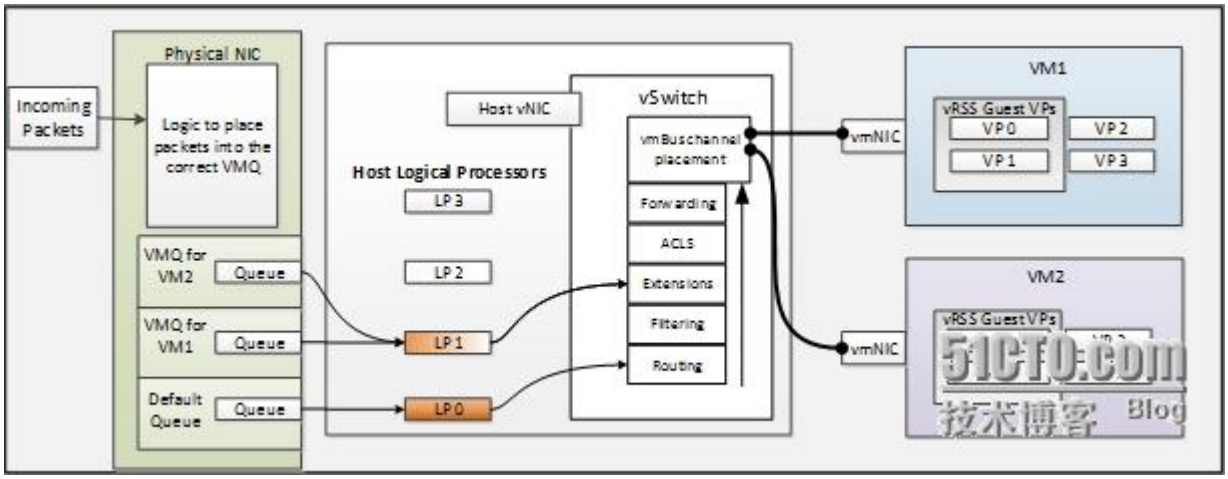
在此之前还是想多说一说有关 VMQ 的概念性问题，因为在我看来，实际体验一项功能之前，十分有必要去了解它的运行机制和工作原理以及关键的要点。

在 Windows Server 2012 中，VMQ 变得更加智能化，微软称之为动态 VMQ，也就是如下图所示：

原来 LP1 与 LP2 是分别处理 VM1 与 VM2 负载的。

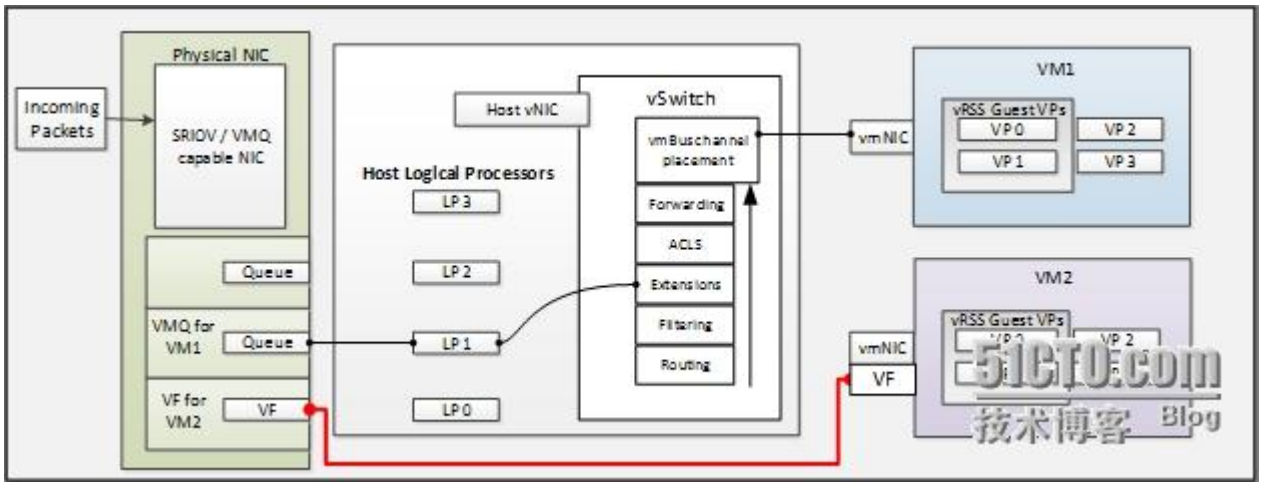


当网络负载变小的时候，VMQ 会合并队列，将 VM1 与 VM2 的负载整合到 LP1 上去 run。



##

上面提到的是动态 VMQ，而当用户想将 VMQ 与 SRIOV 复用时，就需要注意了——鱼与熊掌不可兼得，为什么呢？看下图：

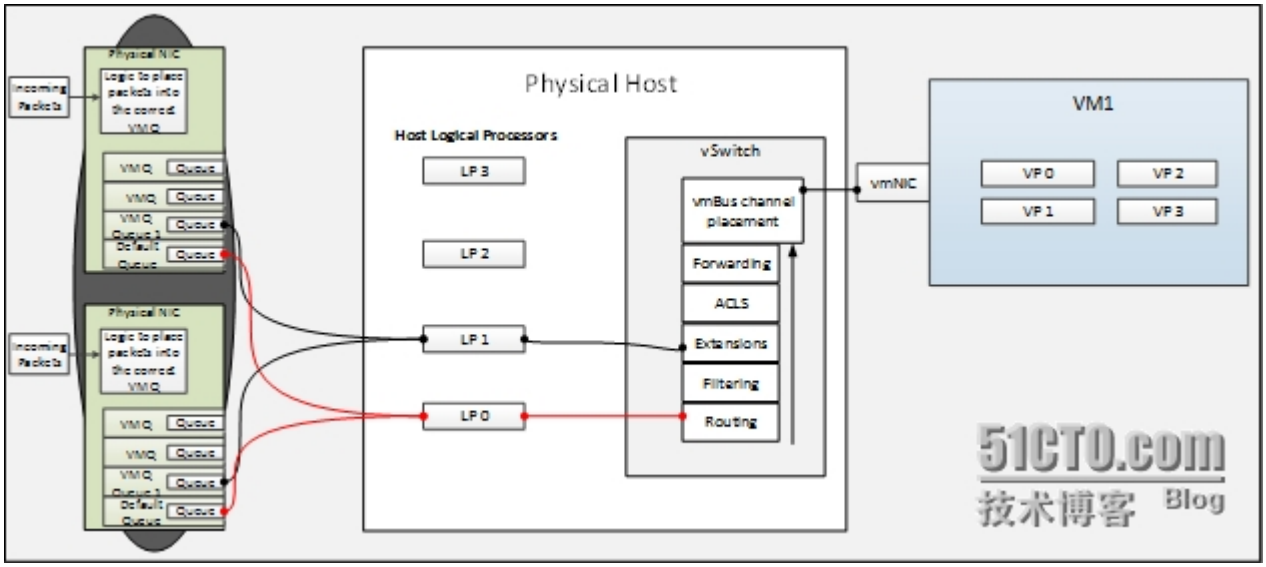


之前的博文提到过（点击开头传送门），SRIOV 通过物理 NIC 实现 VF 功能，使得网络流量越过虚拟化堆栈，也就是跳过了虚拟交换机这一环，特别是 extension 这一项，那么 VMQ 的存在就没有意义，因此 SRIOV 与 VMQ 只能二选其一，这个要视用户的场景来取舍了。

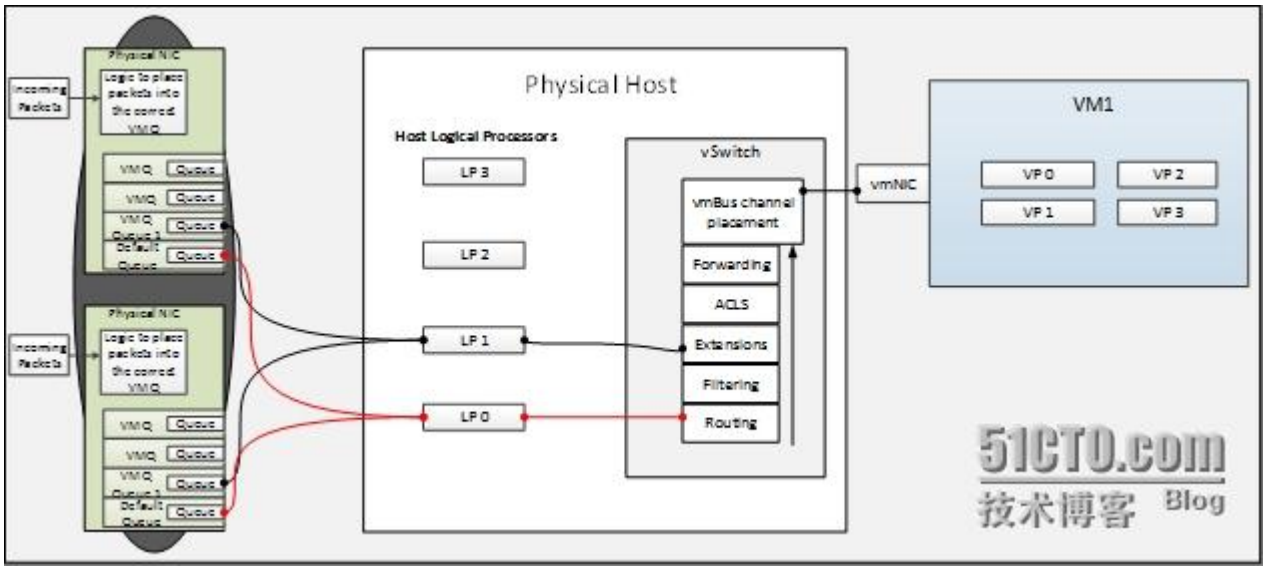
##

还有一种场景需要特别提及一下，就是 VMQ 与 NIC Teaming 配合使用时，需要注意以下问题：

1. NIC Teaming 有多重模式和算法，当使用交换机依赖模式时（switch dependent），无论负载平衡算法是地址哈希（address hash）、动态（dynamic）还是 Hyper 端口（Hyper-V port），那么此时 VMQ 都将处于一种叫做“min queues”的模式下，如下图：

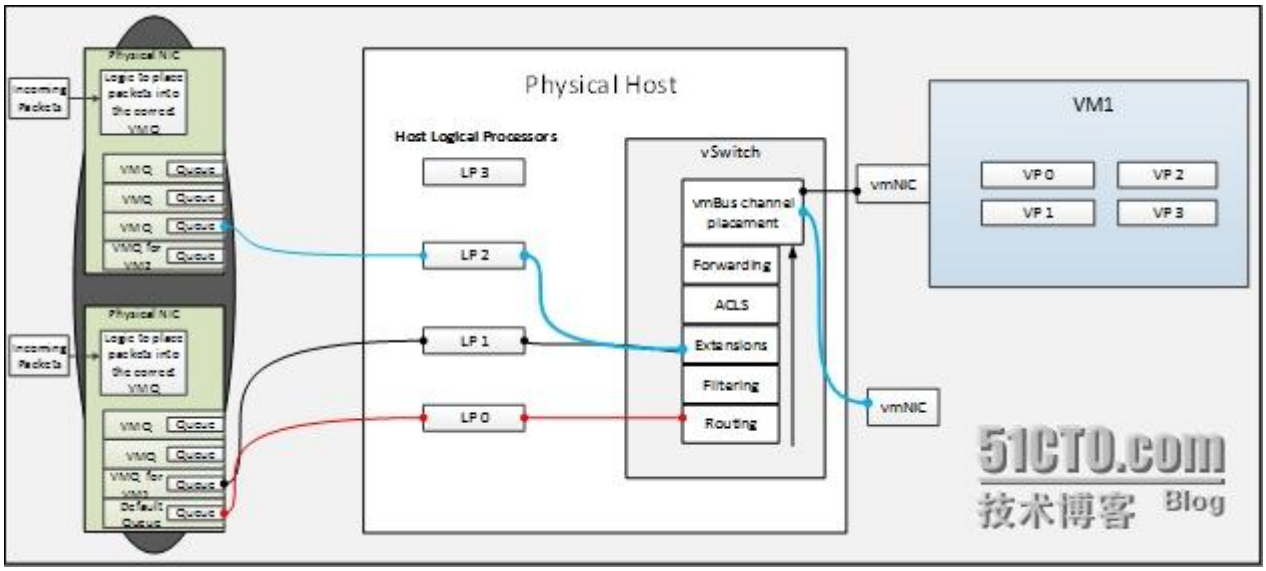


这种模式的特点是什么呢，因为入方向流量不固定，也就是这次可能走 NIC1 进来，下次可能就从 NIC2 进来了，那么 VMQ 的队列数就无法最大化利用，因为需要确保针对 VM1 的队列，在 NIC1 和 NIC2 上都要一致，比如在两块 NIC 上都需要复用同一颗 LP，简单的理解就相当于一个 mirror 模式。



2. 当使用非交换机依赖模式时（switch independ），只有当算法选用地址哈希（address hash）时会处于“min queues”，其它两种算法都会变更为“sum of queues”模式，如下图：

这种模式下就不会出现 mirror 的效果，NIC 队列数量将会最大化的被利用，因为此时入方向流量是固定的，只会出现在一块 NIC 上。



那么根据上面的说明，就可以看出，在 NIC Teaming 模式下启用 VMQ，不同的算法会导致不同的性能结果，那么就需要针对 VMQ 多做一些额外配置，此外还需特别注意：

1. VMQ 针对于主机 host 或某一台虚机，有且只能有一颗 LP 来承载，不可能超过一颗。
2. 在开启了超线程后，VMQ 只在偶数节点上起作用，因此通常建议用户把“HT”（hyper-threading）功能关掉。

接下来看个示例，假设我的物理服务器有 8 核（即 8LP），两块 NIC 组成 Teaming 模式，在 min queues 与 sum of queues 两种模式下配置会有所区别：

1. min queues

   “Set-NetAdapterVMQ –Name NIC1, NIC2 –BaseProcessorNumber 0 –MaxProcessors 8”（两块 NIC 都需要调用 LP0~LP8，它们是复用的）

2. sum of queues

   “Set-NetAdapterVMQ –Name NIC1 –BaseProcessorNumber 0 –MaxProcessors 4”（NIC1 将调用 LP0~LP3）

   “Set-NetAdapterVMQ –Name NIC2 –BaseProcessorNumber 4 –MaxProcessors 4”（NIC2 将调用 LP4~LP7）

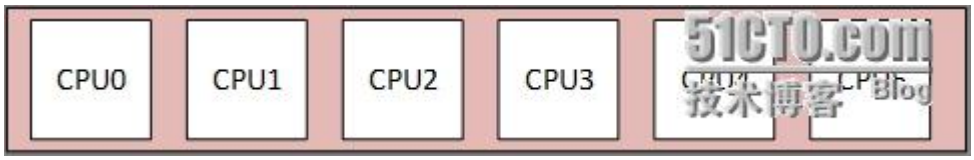
这里希望不要把大家绕晕，其实有兴趣的童鞋可以自己用 PowerShell 看一看 VMQ 的相关 command 以及一些重要属性，下面几个图示比较清晰的阐述了这些参数之间的关系：

1. 以 6 颗 LP 的主机为例（LP0~LP5，抱歉水印挡住了视线），它们的关系是这样的：

   baseprocessornumber：0

   maxprocessornumber：5

   maxprocessor：6

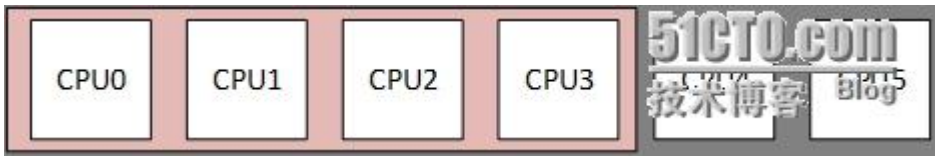


2. 当“可被调用的最大 LP 号”发生变化时，只有 LP0~LP3 可以被使用，即使“最大处理器数量”仍然是 6，但是优先级也要服从于“maxprocessornumber”。

   baseprocessornumber：0

   maxprocessornumber：3

   maxprocessor：6

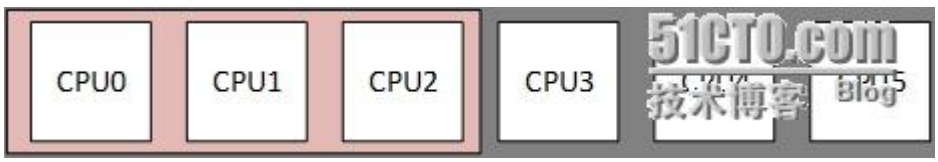


3. 当“最大处理器数”发生变化时，虽然“maxprocessornumber”是 5，但是此时“maxprocessor”值的优先级更高

   baseprocessornumber：0

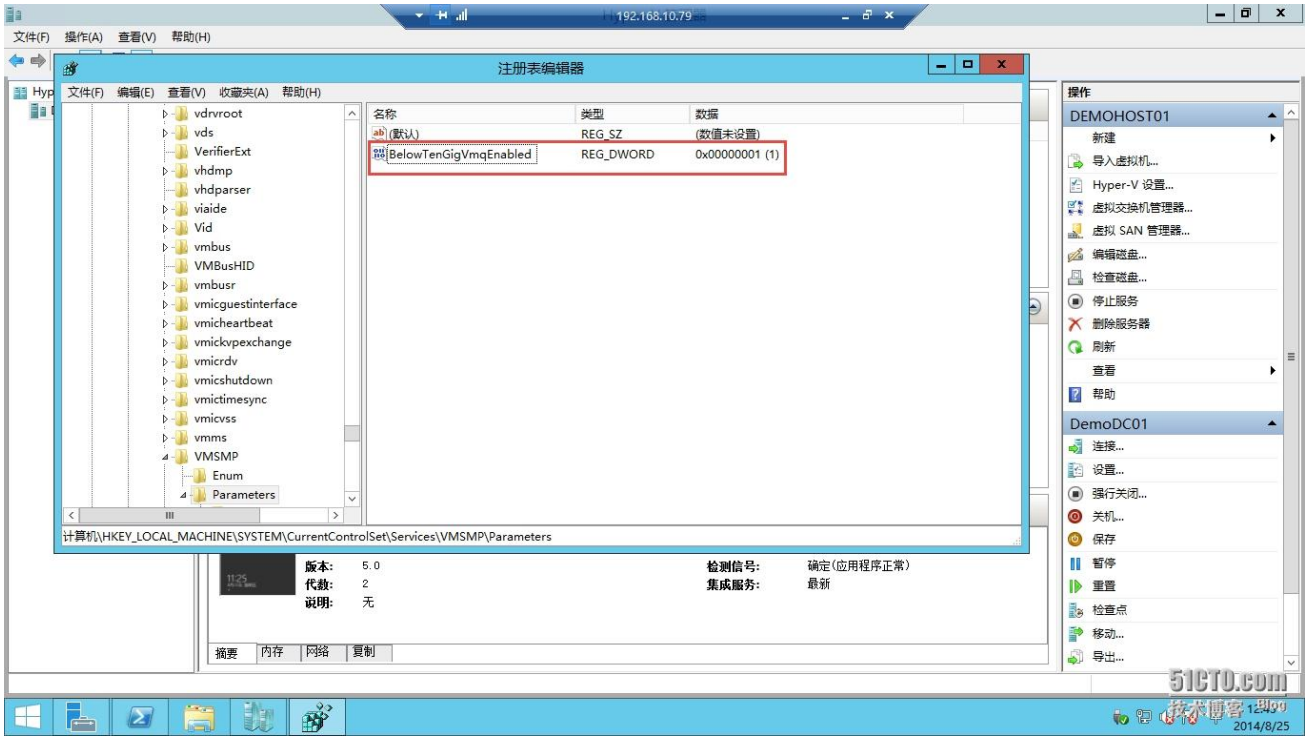
   maxprocessornumber：5

   maxprocessor：3



##

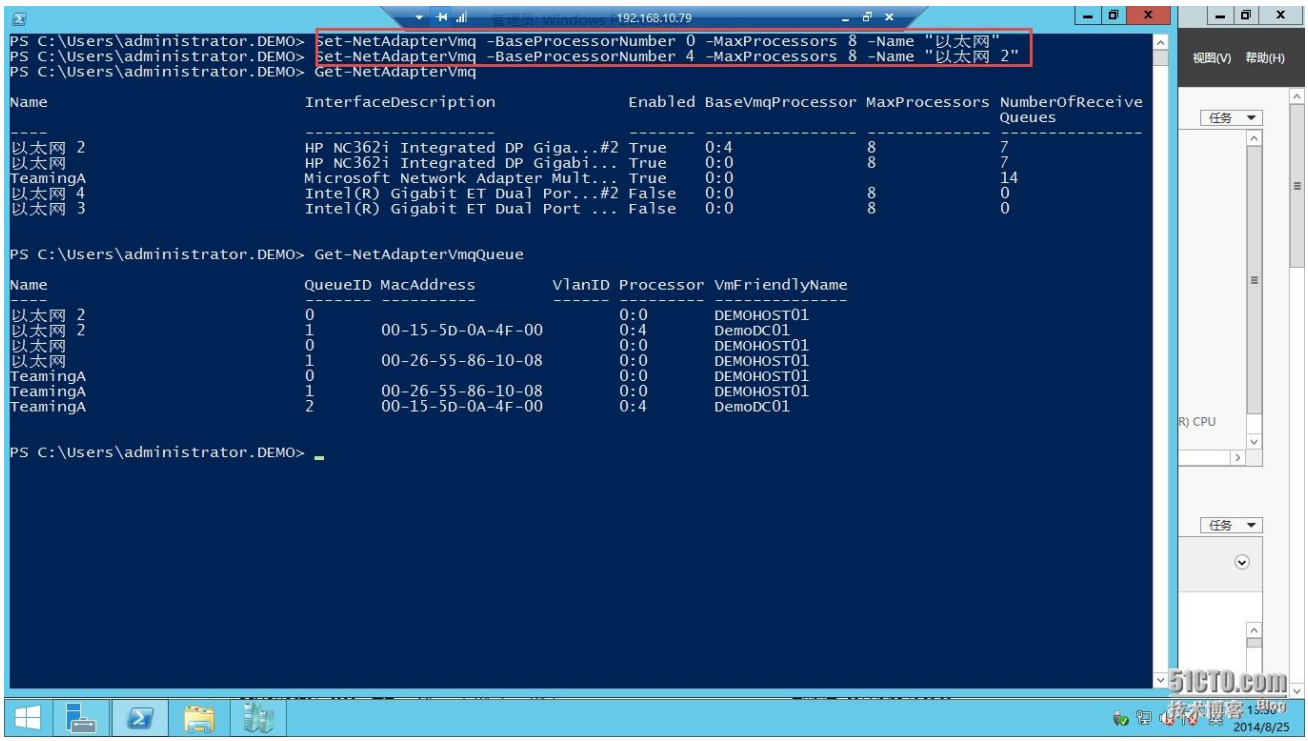
下面来看一下测试环境中 VMQ 的实际效果，因为我的屌丝环境没有 10Gi 网卡，只能拿 1Gi 的凑合了，然而操作系统默认是不会启用千兆环境下的 VMQ 功能的，所以我要先修改一***册表，在“HKLM\system\currentcontrolset\services\VMSMP\parameter”下添加一行 dword 值“BelowTenGigVmqEnable”并赋值为“1”，然后重启物理机就可以了，如下图：



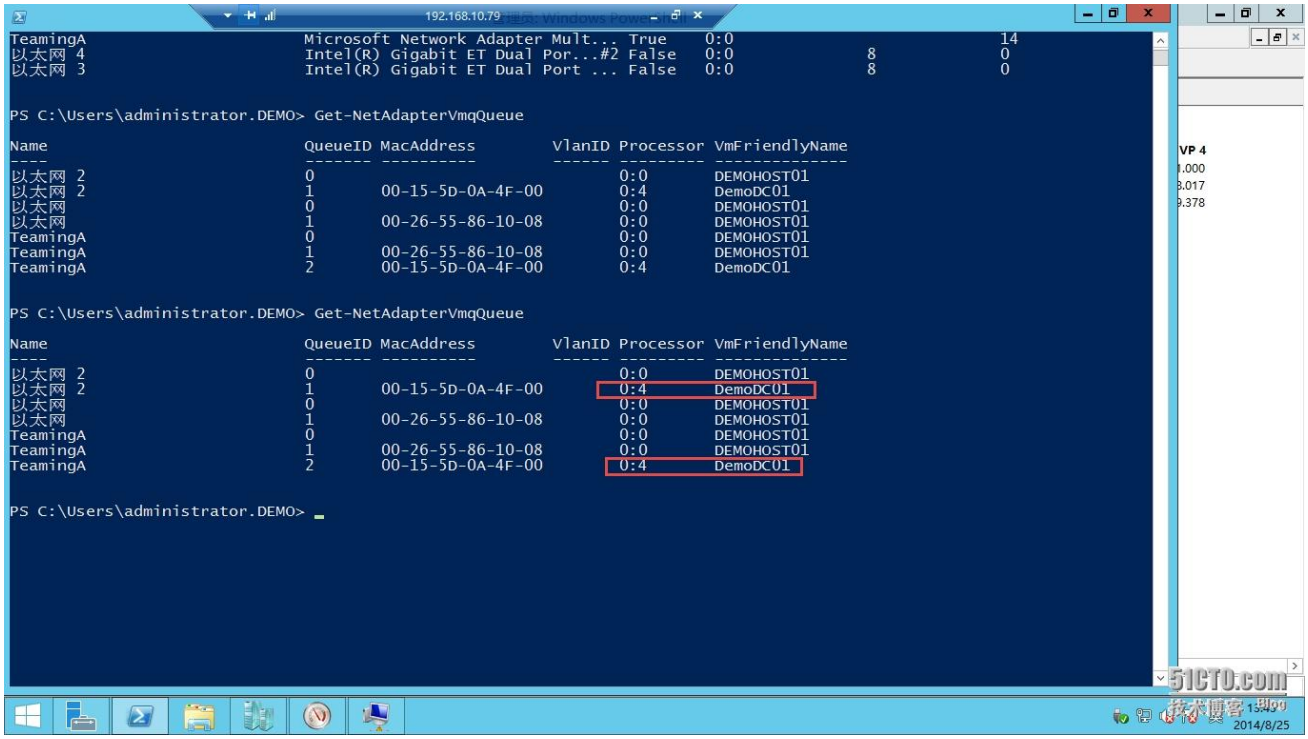
接下来通过 PowerShell 查看当前物理 NIC 的 VMQ 属性，以我的环境为例，以太网和以太网 2 都是启用的，并且因为我的 Teaming 模式是 switch independent 以及 dynamic 算法，所以当前处于“sum of queues”模式，因此根据物理机条件（8 核）进行配置，将以太网绑定到 LP0~LP3 上，以太网 2 绑定到 LP4~LP7 上，每个 NIC 各 4 条通道，之后再通过 PowerShell 查看 VMQ 队列分配情况，如下图：

1. 以太网和以太网 2 都有一条队列 ID 为 0 的数据，它们是分配给默认队列的，所谓默认队列就是既不属于某台 VM 也不属于主机 host 的流量将被转发到默认队列中进行处理。
2. 在以太网中，队列 ID 为 1 的条目已经被分配给了 MAC 地址为“00-26-55-86-10-08”的主机 host。

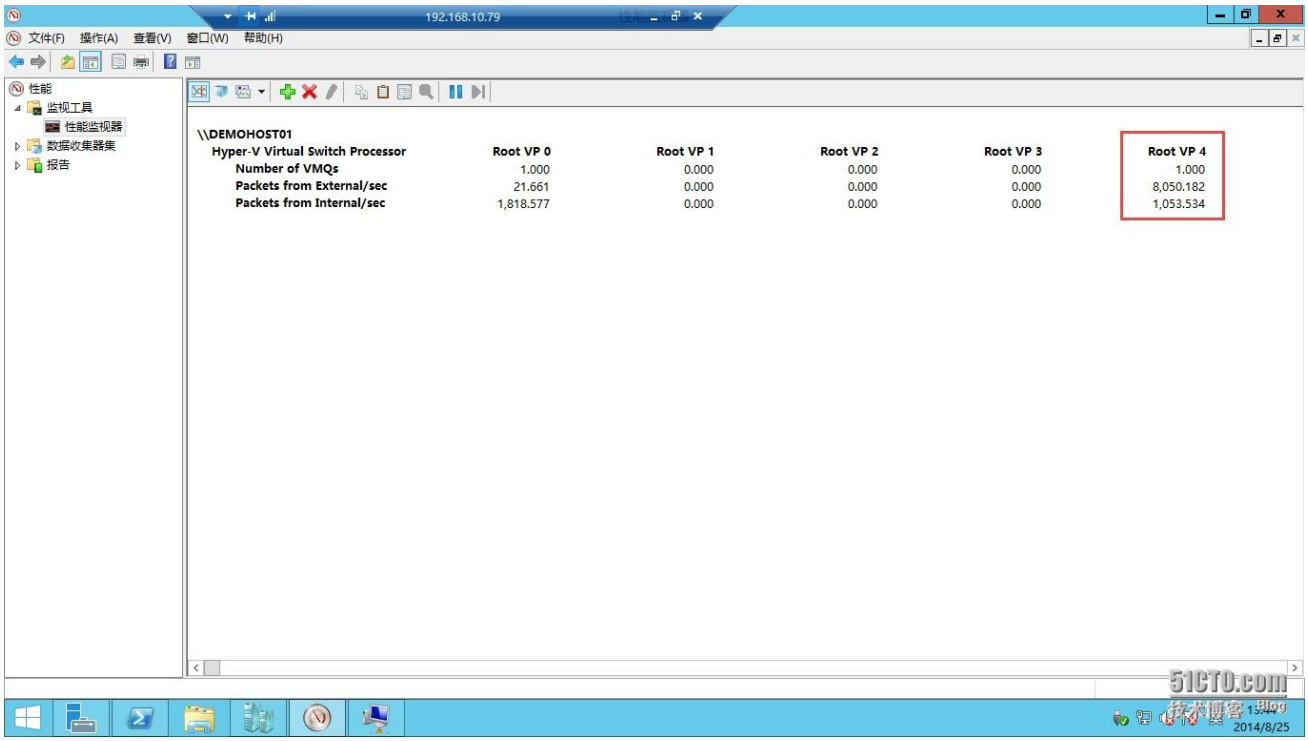
3. 在以太网 2 中，队列 ID 为 1 的条目已经被分配给了 MAC 地址为“00-15-5D-0A-4F-00”的虚拟机 demoDC。



而下面这张图显示了，当前是由 LP4 来处理 demoDC 这台虚机的流量负载的。如下图：



通过主机的性能计数器可以清楚的看到，目前 demoDC 正在通过网络拷贝文件，负载都在 LP4 上



##

RSS 与 VMQ 这两项硬件加速技术对于云计算平台的性能改进是巨大的，特别是在万兆网络条件下，用户肯定是希望能够资源利用最大化的，此外除了网络，在存储方面也有一些辅助功能，例如 ODX，当然我想自己恐怕是找不到支持的硬件了:(，等以后有机会再说吧。

部署 Lync Server 2013 SQL 见证服务器

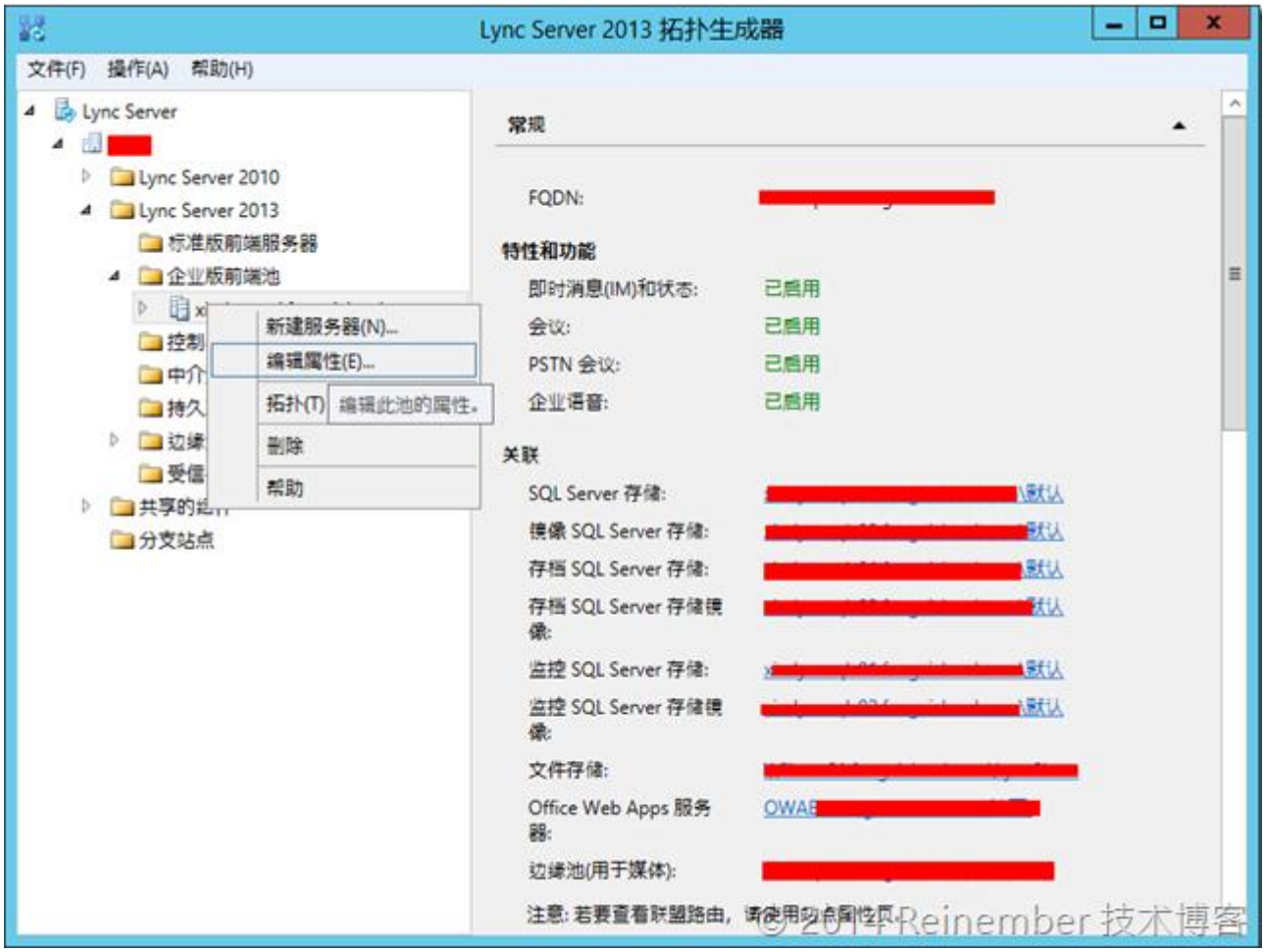
作者；reinxu 来源：http://reinember.blog.51cto.com/2919431/1557275

最近由于某些需要，给 Lync 部署了一套基于 SQL 镜像的高可用后端，今天就和大家分享一下，后端镜像的见证服务器部署。首先，我们在 SQL 的见证服务器上安装 SQL Express 版，建议版本与 SQL 主服务器的版本一样，但 Express 版就足够了。

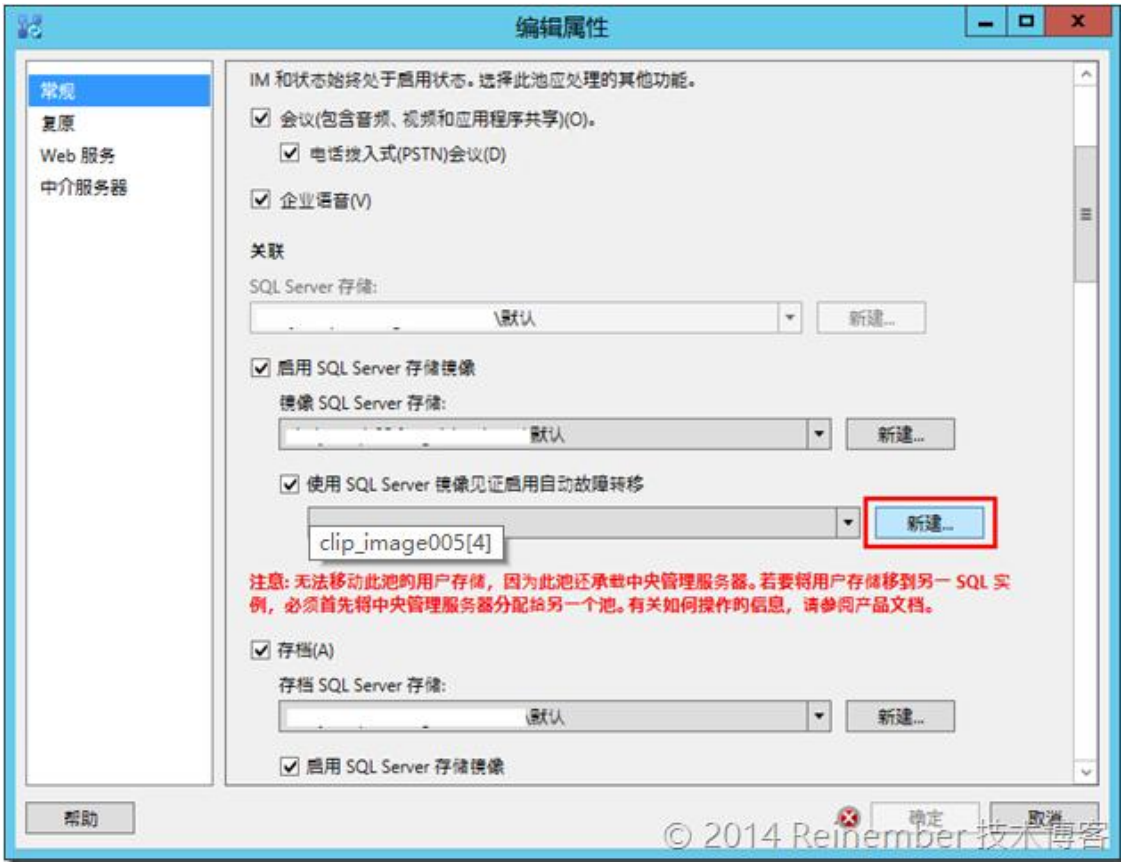
在安装 SQL 过程中我们需要设置 SQL Server 服务的服务帐户，需要指定域帐户。

然后将我们事先创建好的 SQL 管理员加到 SQL 管理员列表中。

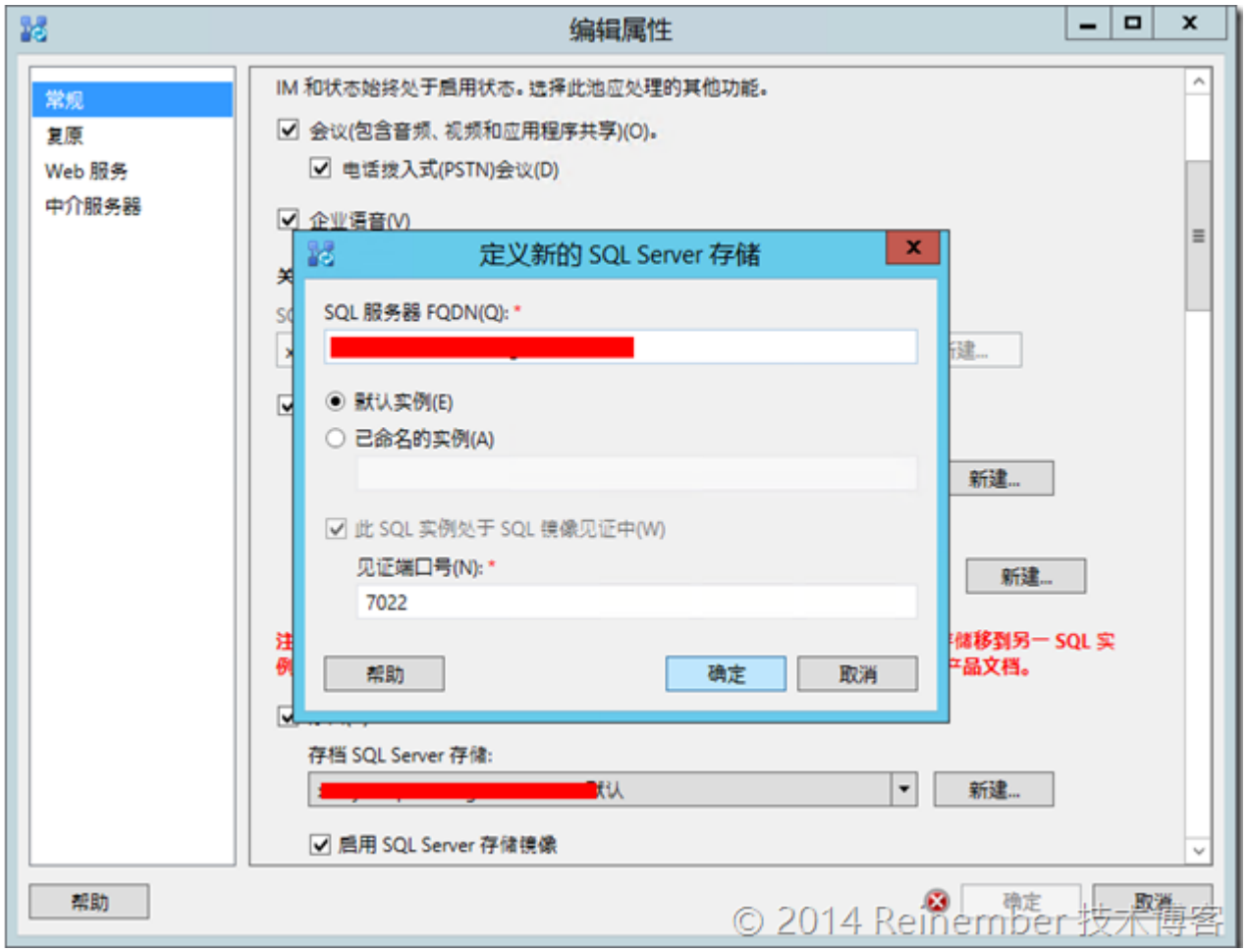
完成安装后，我们回到 Lync Server 拓扑生成器中，编辑企业版前端服务器池的属性。



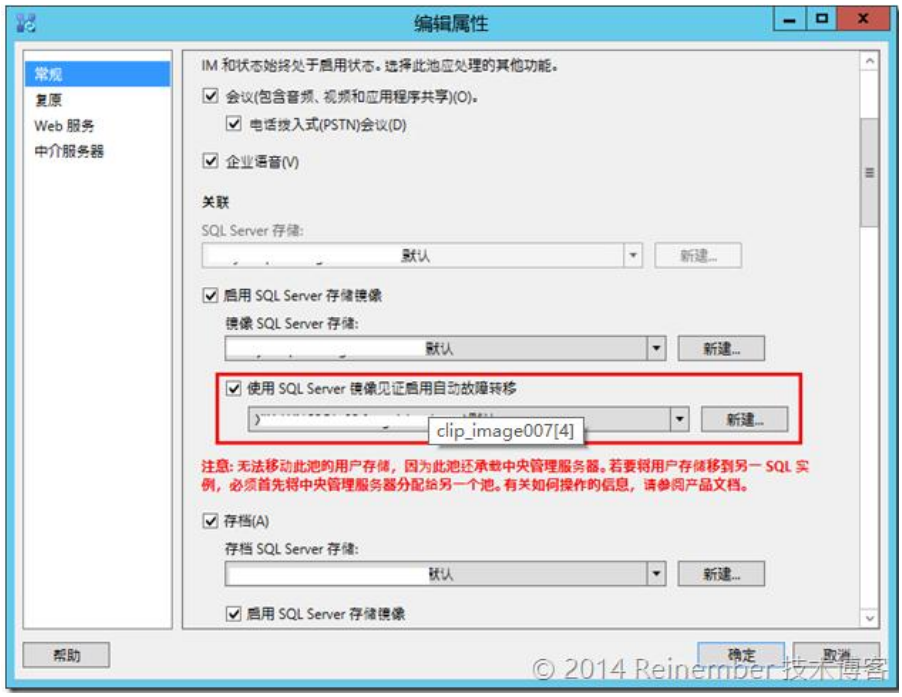
在关联位置，复选 SQL 镜像见证，并单击右侧的新建。



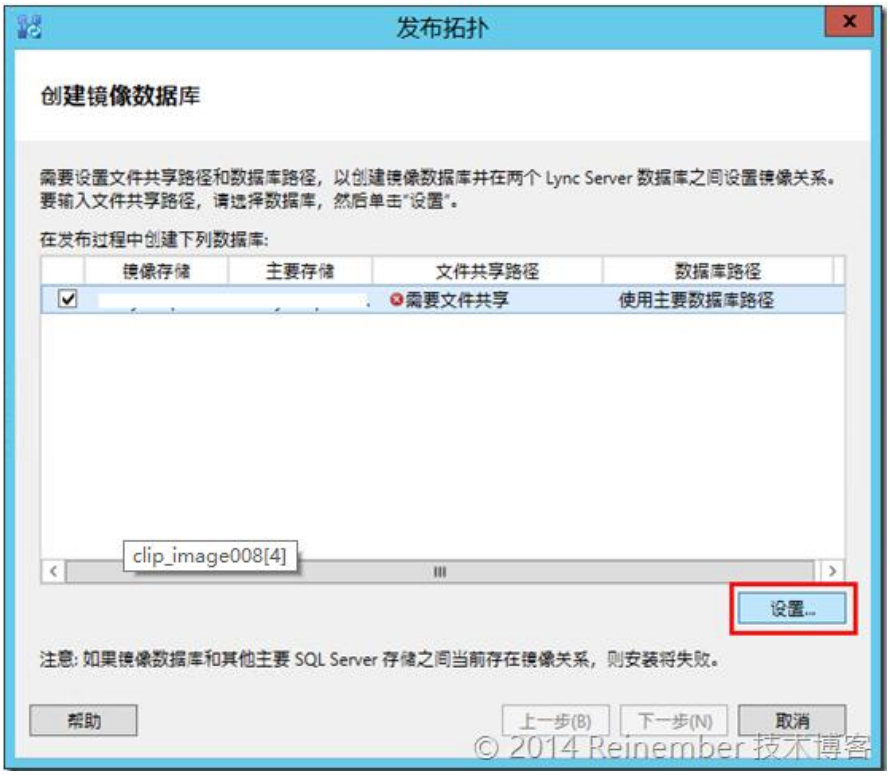
输入我们 SQL 见证服务器的 FQDN，以及配字实例名称，如果是默认则不需要修改，并配置端口号（使用默认即可）。



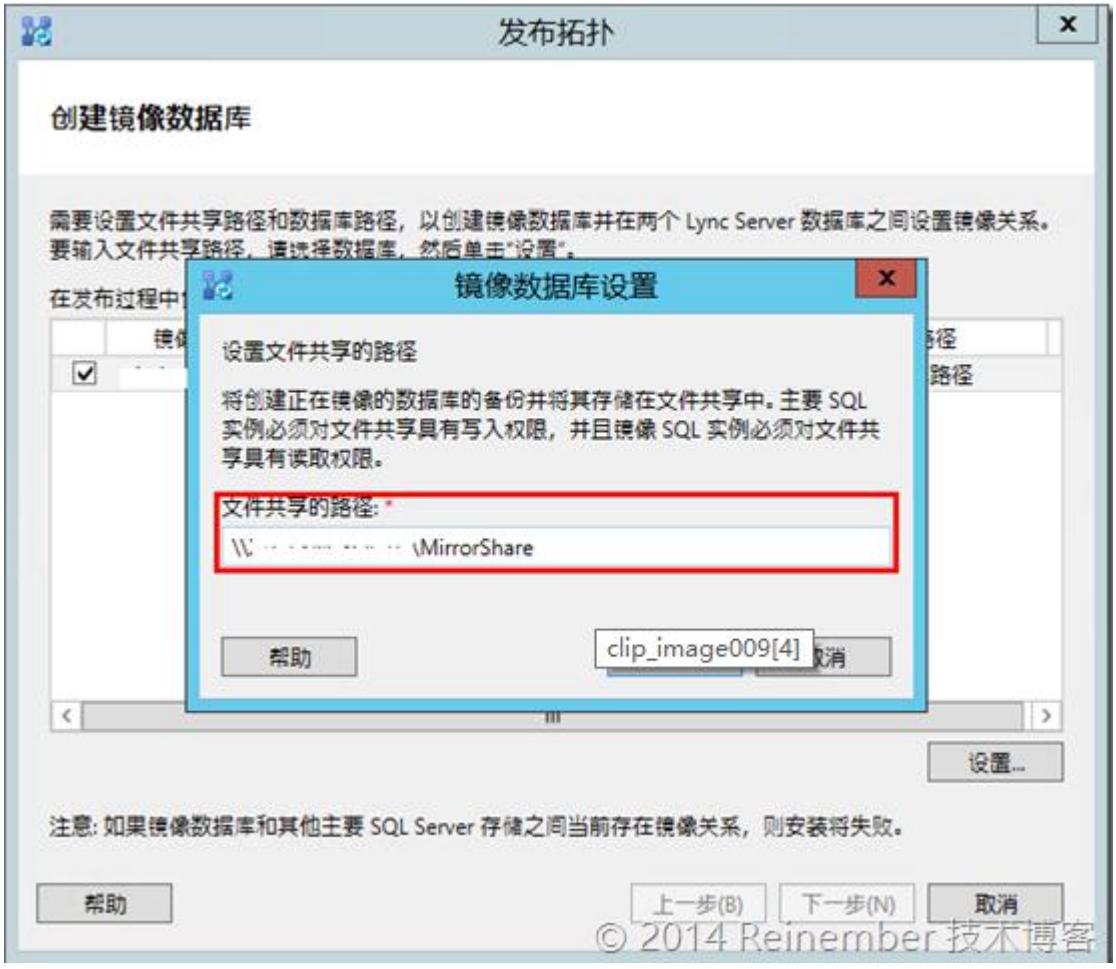
完成之后，可以在编辑属性对话框中看到我们配置的见证服务器。



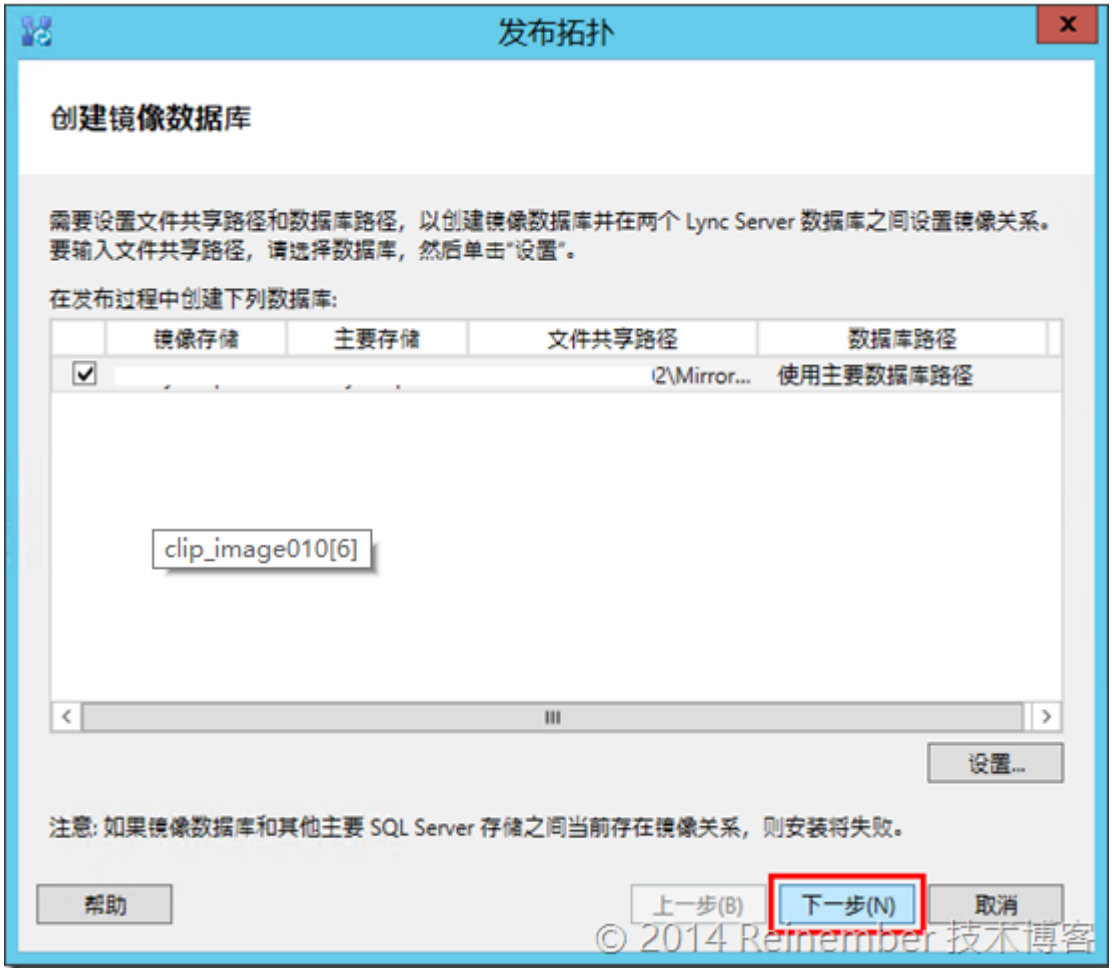
然后我们在发布拓扑时需要指定文件共享路径的位置。



我们这里根据实际情况设置即可。



回到发布拓扑界面，我们即可单击下一步进行拓扑发布了。



Lync SQL 数据库镜像及见证服务器还是蛮有意思，整体的思路是很清晰的，就是 SQL 的镜像模式，只不过 Lync Server 可以直接去调用部署，而不需要手动的去操作太多 SQL 的东西。很久没有写东西了，也许是最近太累了，越大越烦心，但不管怎样还是应该踏踏实实的去做，是吧。大家如果有什么问题或疑问，欢迎留言，我会尽量的与大家分享我所了解的。

华为 RH5885H v3 服务器 RAID 设置及问题解析

作者：孙杰 来源：http://xjsunjie.blog.51cto.com/999372/1551823

今年春，华为全球首发基于英特尔至强 E7 v2 处理器的系列服务器新品，其中包括 RH8100 V3 八路服务器、RH5885H V3 四路服务器和 E9000 刀片服务器的四路计算节点 CH242 V3 服务器。最近单位新购了几台华为 RH5885H V3 四路服务器，准备做测试用。在装 RHEL 系统前，一般需要做 RAID,下面就这款服务器的 RAID 设置及相关问题进行解析。

服务器外型如图所示：

开机启动 RH5885H V3，看到的画面如下：

在这个界面闪过后，当你看到提示“Press <Ctrl><R> to Run MegaRAID Configuration Utility”提示信息时，按“Ctrl+R”键。进入“SAS3108BIOS Configuration Utility”界面，如下图所示。

显示有 8 块盘，每盘 300G。

每个菜单项说明如下:

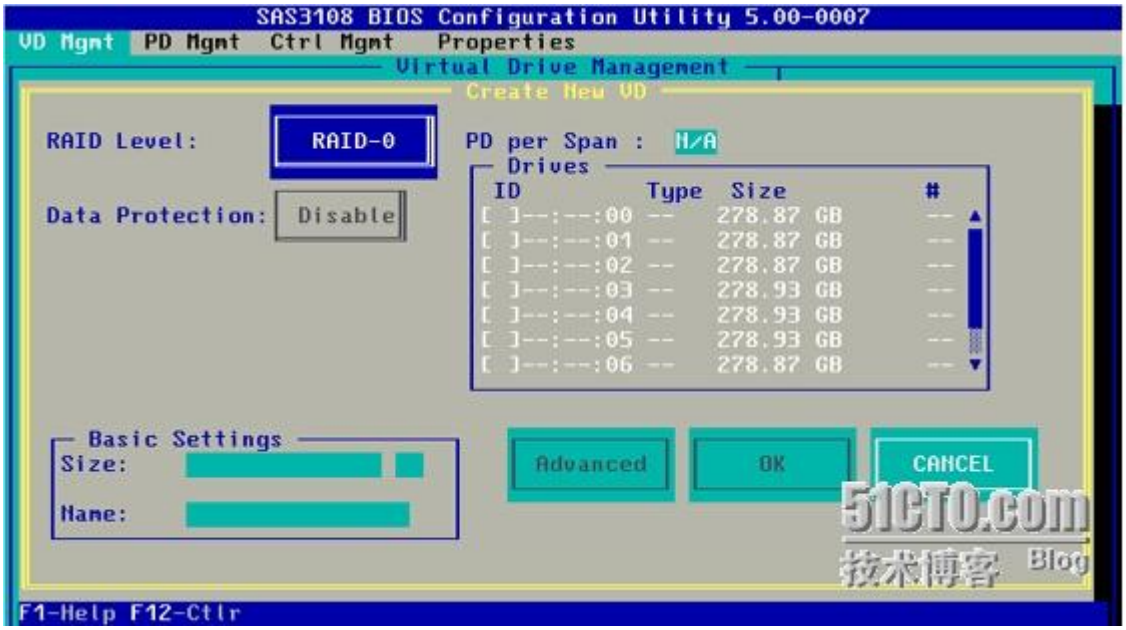
VD Mgmt	虚拟磁盘管理界面。
PD Mgmt	物理磁盘管理界面。
Ctrl Mgmt	控制器管理界面。
Properties	RAID 卡属性查询界面。
Foreign View	外部配置管理界面。

以配置 RAID5 为例

创建的 RAID 5 最少由 3 块硬盘组成。

在如上界面按“F2”键，在弹出的列表中选择“Create Virtual Drive”。

打开如下所示界面。



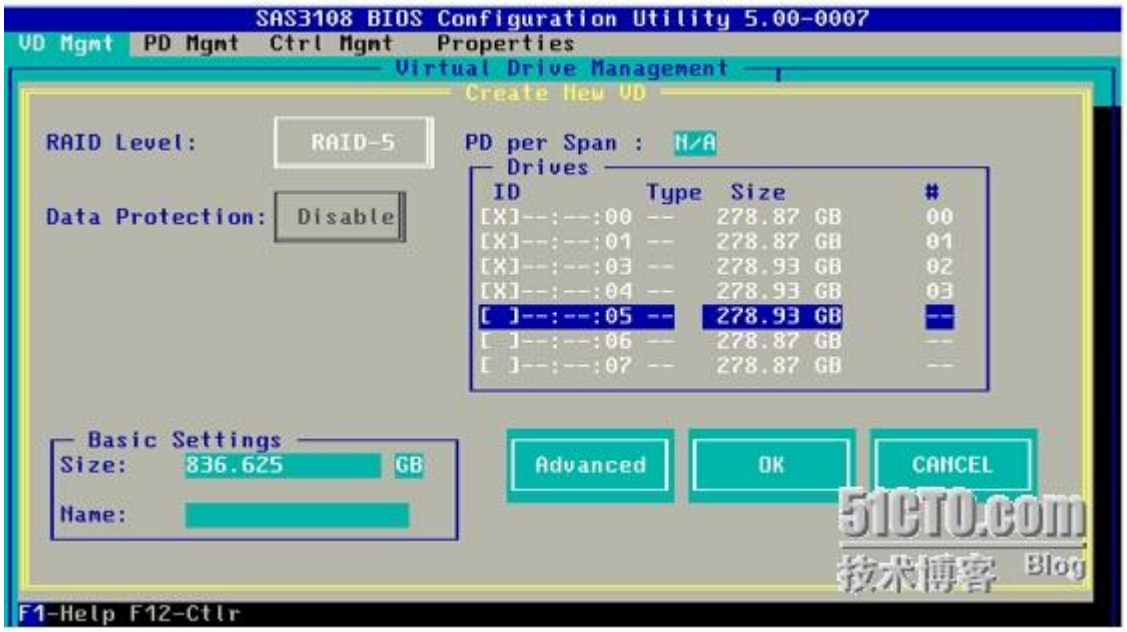
设置 RAID 级别为 RAID 5

在“RAID Level”区域框按“Enter”，并通过“↑”、“↓”选择 RAID 级别为“RAID 5”。

添加硬盘

按 TAB 键将光标迁移到“Drives”区域。

按“↑”、“↓”移动光标，按“Enter”选择要添加到 RAID 组的硬盘，选中硬盘的“ID”会显示为“[X]”。如图 5-20 所示



设置 RAID 容量和名称

按“↓”将光标迁移到“Basic Settings”区域。

光标移至“Size”区域，根据实际需要设置 RAID 容量。

不设置时，系统采用当前 RAID 支持的最大容量作为“Size”的默认值。

光标移至“Name”区域，设置 RAID 名称。

设置高级属性

选中“Advanced”并按“Enter”。

打开 RAID 高级属性设置界面，如图 5-21 所示。

高级属性中的参数说明如下表。

参数项	说明
Strip Size	每个硬盘上的数据条带的大小。默认配置为 256KB。
Read Policy	虚拟磁盘的数据读策略，分以下三种： l Normal：关闭预读取功能（Read Ahead）。 l Ahead：使能预读取功能。控制器可以预读取顺序数据或预测需要即将使用到的数据并存储在 Cache 中。
Write Policy	虚拟磁盘的数据写策略，分以下三种： l Write Back：当控制器 Cache 收到所有的传输数据后，将给主机返回数据传输完成信号。该设置为推荐设置的标准模式。 l Write Through：当磁盘子系统接受到所有传输数据后，控制器将给主机返回数据传输完成信号。 l Write Back with BBU：在控制器无 BBU（Battery Backup Unit）或 BBU 损坏的情况下，控制器将自动切换到 Write Through 模式。
I/O Policy	应用于特殊的虚拟磁盘读取，不影响预读取 Cache。分以下两种： l Direct：直接读取并不在 Cache 中缓存。 l Cached：所有的读取在 Cache 中缓存。
Disk cache Policy	磁盘 Cache 策略： l Unchanged：保持当前磁盘 cache 策略。 l Enable：使能磁盘 cache。

参数项	说明
	l Disable：禁止磁盘 cache。

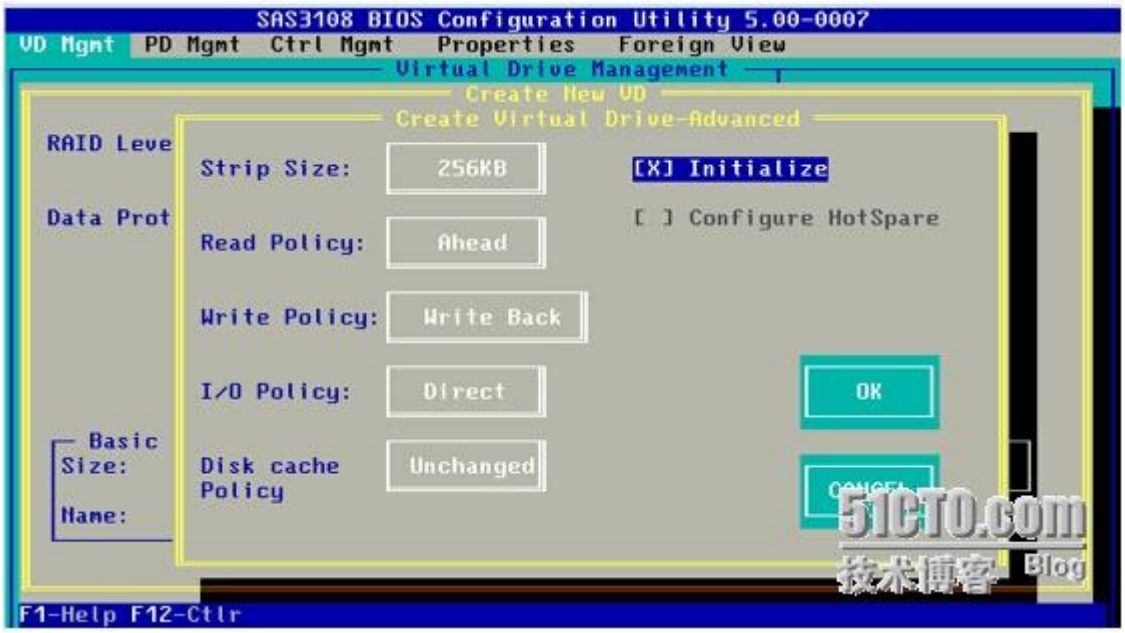
按照实际情况设置 RAID 的高级属性参数。

然后按“↑”或“↓”移动光标，按“Enter”选择“Initialize”。

选中该选项后，RAID 创建完成时自动进行初始化操作。

在此页面的高级对话框中单击“OK”。

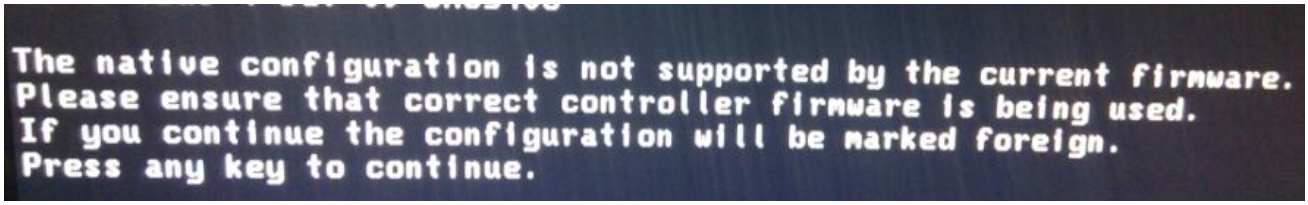
选中“Initialize”后，会在前方显示“[X]”，如下图所示。



在返回的“Create New VD”界面中选择“OK”并按“Enter”。这样 RAID5 就创建完毕了，重启后放入系统光盘就可以安装系统了。

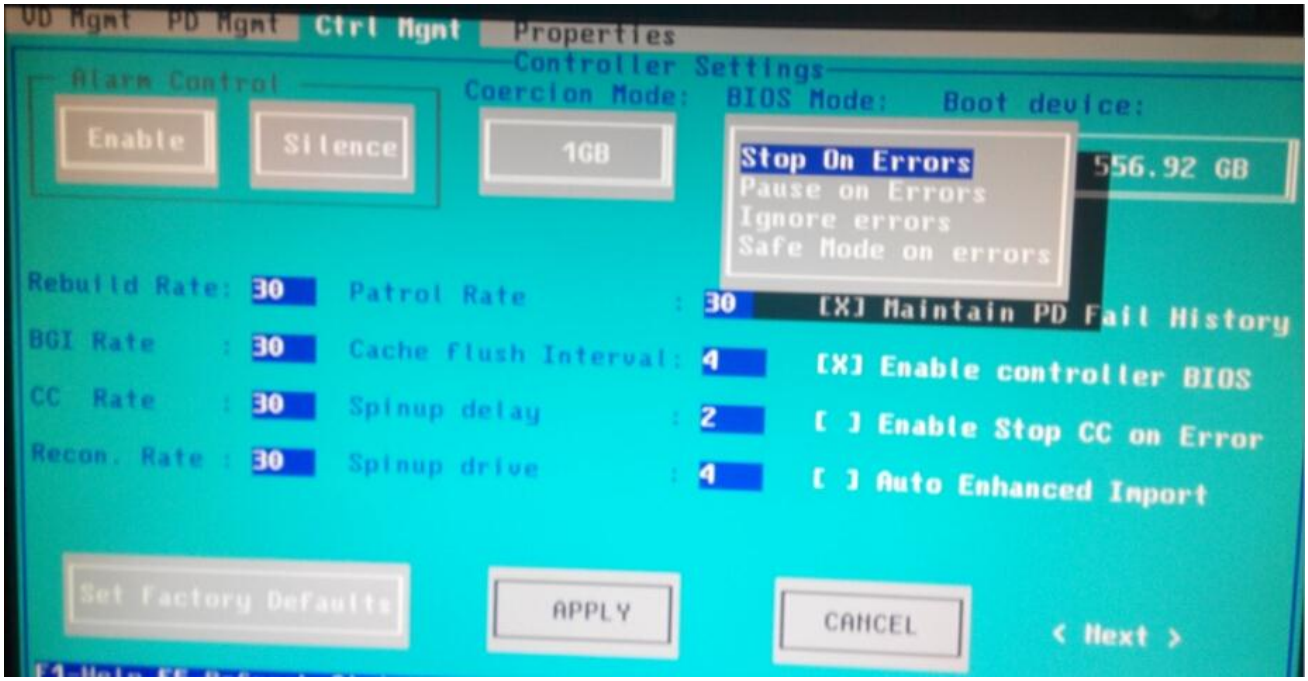
RAID 设置完毕后，在重启容易遇见的一个报错及解决

错误信息：



这是因为 RAID 卡检测到与卡上原有的 RAID 信息不匹配，该 SAS 3108 的 RAID 卡之前在其他平台使用过。这样当你更换到另一个平台，就会在启动时报如下错误，且启动会停住。要解决该问题，只需要简单设置一下即可。

进入当初配置 RAID 的界面，切换到 ctrl mgmt 界面，在 BIOS MODE 模式里将 STOP ON ERRORS 改成 PAUSE ON ERRORS 或 IGNORE ERRORS 即可。这样就不会一直停在那个界面了，系统还可以继续启动。



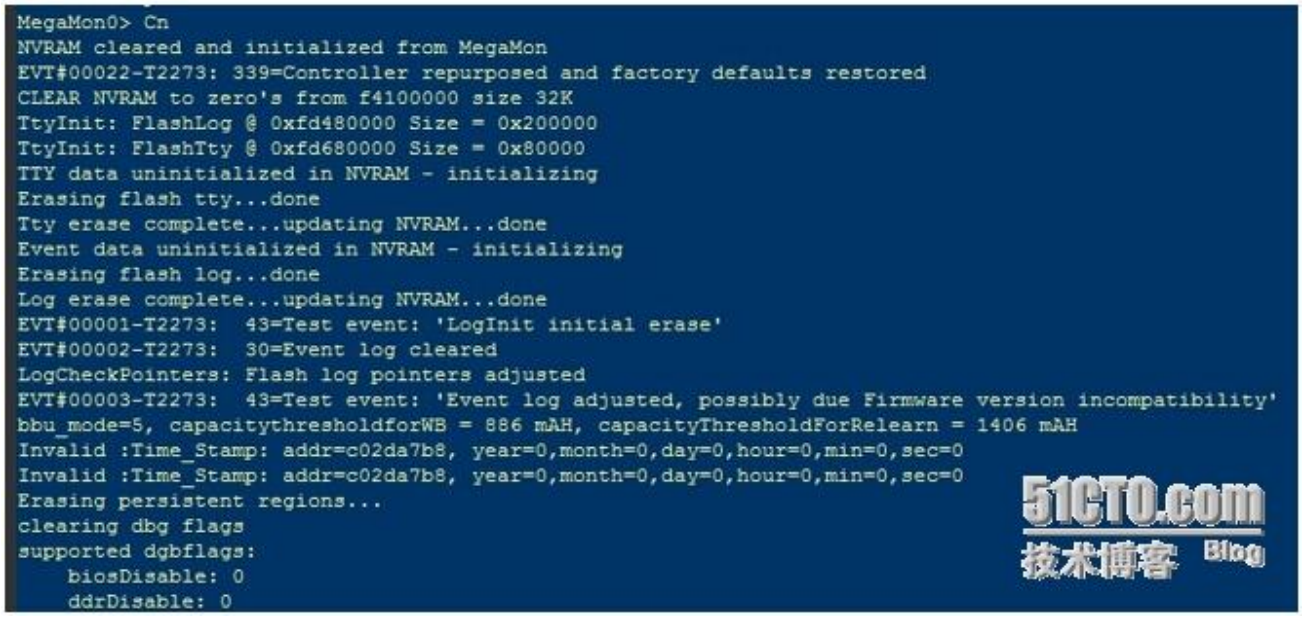
如果要彻底的清除这个消息，则可以这样操作：

预先做好数据备份

(1)连接 3108 的物理串口或通过 SOL 连接 3108 串口皆可

(2)服务器上电，等待 3108 完成初始化至出现”Native Configuration is not supported…”时，在 3108 串口中输入 123m 进入串口命令行 MegaMon：

(3)然后在 MegaMon 中执行 Cn 命令清除 nvram



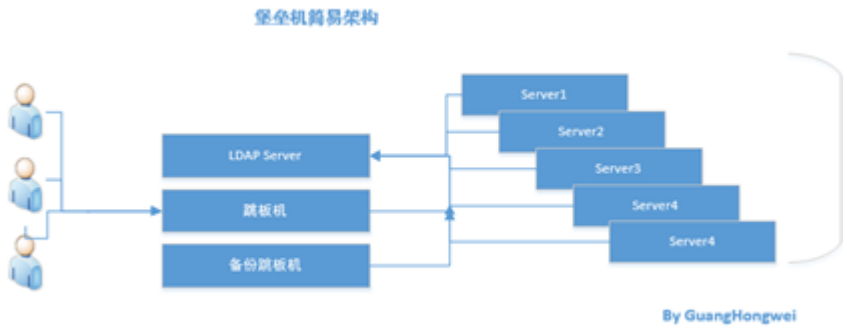
(4)命令执行完后 nvram 就已经完成清除动作。随后可重启服务器正常使用。

开源运维堡垒机(跳板机)系统 python

> 作者：老广 来源：http://laoguang.blog.51cto.com/6013350/1540080

相信各位对堡垒机(跳板机)不陌生，为了保证服务器安全，前面加个堡垒机，所有 ssh 连接都通过堡垒机来完成，堡垒机也需要有 身份认证，授权，访问控制，审计等功能，笔者用 Python 基本实现了上述功能。

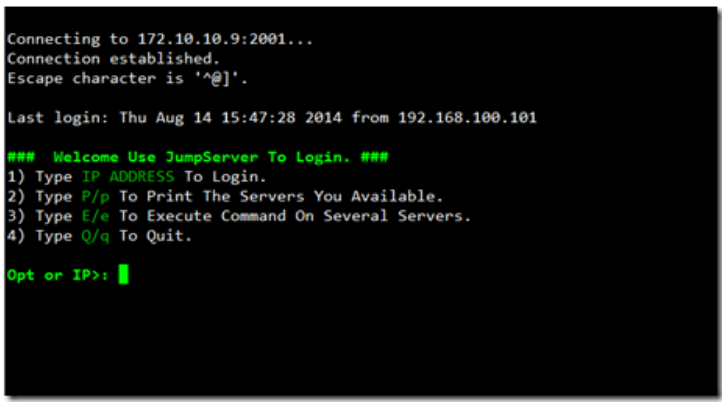
架构：



后端主要技术是 LDAP,配置了 LDAP 集中认证服务器， 所有服务器的认证都是由 ldap 完成的，我的做法是每个用户一个密码，把密码加密放到了数据库中，当用户输入 ip 从跳板机登陆服务器的时候，跳板机系统取出密码，并解密，通过 pexpect 模块将密码发送过去，来完成登录的。

登录界面和方法

用户登录跳板机，用的是秘钥认证，登录跳板机后会自动执行跳板机的系统



输入完整 IP 或者部分 IP 可以完成登录，如果输入的部分 ip 匹配的 ip 不是唯一，会有提示，没有权限的会提示没有权限

```
###  Welcome Use JumpServer To Login. ###
1) Type IP ADDRESS To Login.
2) Type P/p To Print The Servers You Available.
3) Type E/e To Execute Command On Several Servers.
4) Type Q/q To Quit.

Opt or IP>: 53
Connecting 172.16.0.53 ...
Login 172.16.0.53 success!

Last login: Thu Aug 14 15:52:57 2014 from 172.10.10.9
[baidutest@software ~]$
```

```
###  Welcome Use JumpServer To Login. ###
1) Type IP ADDRESS To Login.
2) Type P/p To Print The Servers You Available.
3) Type E/e To Execute Command On Several Servers.
4) Type Q/q To Quit.

Opt or IP>: 172
172.16.0.53
172.16.2.88
172.16.2.36
172.16.2.36
Opt or IP>:
```

输入 P/p 可以查看自己拥有权限的服务器 ip

```
###  Welcome Use JumpServer To Login. ###
1) Type IP ADDRESS To Login.
2) Type P/p To Print The Servers You Available.
3) Type E/e To Execute Command On Several Servers.
4) Type Q/q To Quit.

Opt or IP>: p
172.16.0.53
172.16.2.88
172.16.2.36 -- hello world
172.16.2.36 -- hello world
Opt or IP>:
```

输入 E/E 可以在几台服务器上执行同样的命令, IP 直接以逗号分隔

```
3) Type E/e To Execute Command On Several Servers.
4) Type Q/q To Quit.

Opt or IP>: e

Input the Host IP(s),Separated by Commas, q/Q to Quit.

ip(s)>: 172.16.0.53,172.16.2.88

Input the Command , The command will be Execute on servers, q/Q to quit.

Cmd(s): date
############### 172.16.2.88 ###############

Thu Aug 14 16:00:30 CST 2014

###############  End result  ###############

############### 172.16.0.53 ###############

2014年 08月 14日 星期四 16:01:50 CST

###############  End result  ###############

Cmd(s):
```

日志记录

日志记录用的是 pexpect 自带的日志记录，记录的日志既保存了命令又保存了命令的输出，也不小心讲发送的密码记录(不满意)，pexpect 模块处理有些难做，我的想法是将日志每天再处理一遍，将密码等去掉，日志保存在 logs 目录下面，文件名是 ip_日期_用户名 ps：用的 chinaren 登录的，提示窗口却是 baidutest，这是由于我个人原因导致的。

http://laoguang.blog.51cto.com Free Linux, Share Linux

```
[root@VPN logs]# ls
172.16.0.53_20140808_chinaren  172.16.0.53_20140813_chinaren
172.16.0.53_20140811_chinaren  172.16.0.53_20140814_chinaren
172.16.0.53_20140812_chinaren  172.16.2.88_20140812_chinaren
[root@VPN logs]#
```

```
[root@VPN logs]# tail 172.16.0.53_20140808_chinaren
chinaren@172.16.0.53's password: UKfIhnO4Q9FJpnN

Last login: Fri Aug  8 13:20:06 2014 from 172.10.10.9
[baidutest@software ~]$ echo 'hwllo world'
hwllo world
[baidutest@software ~]$ echo 'hello'
hello
[baidutest@software ~]$ exit
logout
Connection to 172.16.0.53 closed.
[root@VPN logs]#
```

访问控制和授权

访问控制和授权是由一套 web 来实现的

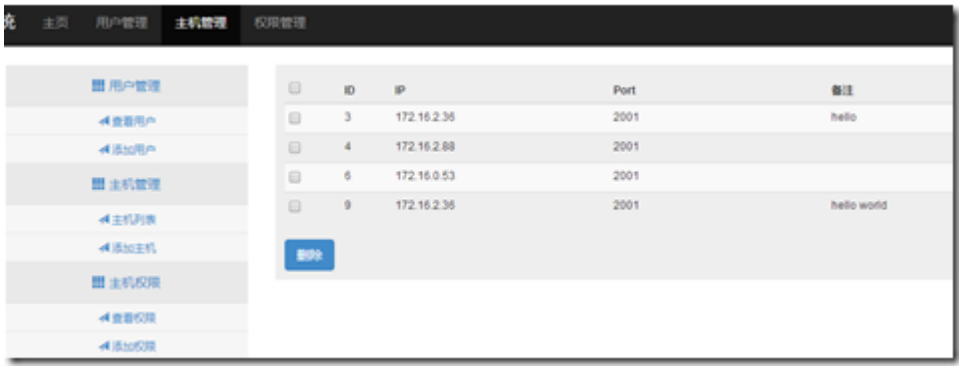
管理员界面

主页:

查看用户：

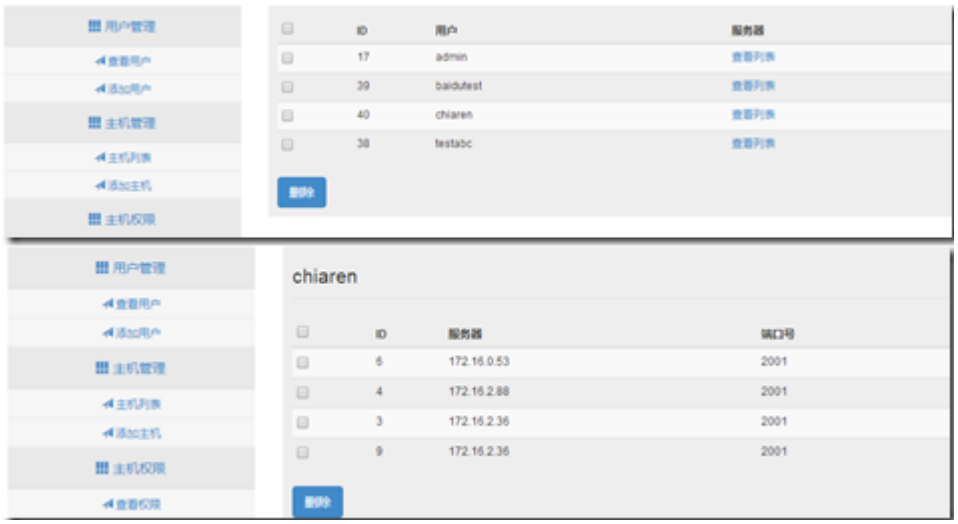
添加用户：

主机列表：

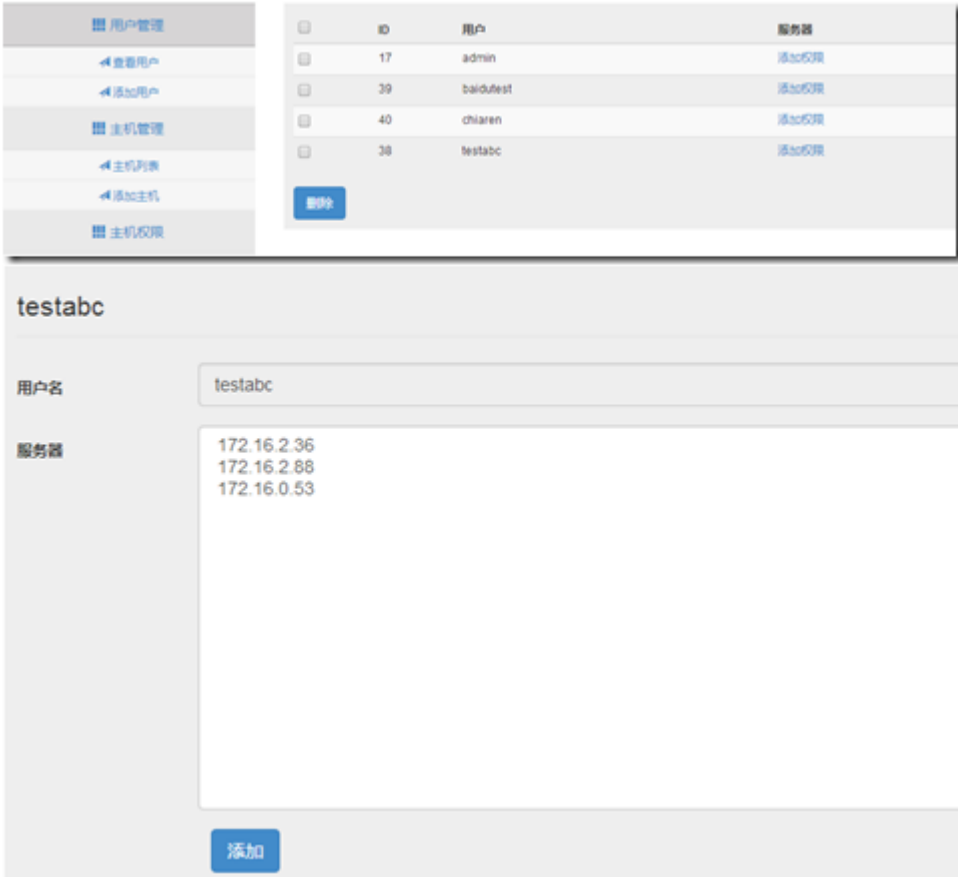


添加主机：

权限列表：



添加权限：



后面的 pptp 和 openvpn 添加是我根据需要添加的，可以去掉

用户登录界面：

更改登录密码：

修改 key 密码：

我把代码放到 github 了，有需要的朋友，可以去看看，大家也可以一同改进，有时间写写部署文档

使用 OWA 重置账户密码后，旧密码依然能够登录问题解决

> 作者：handsome7038 来源：http://lixiaosong.blog.51cto.com/705126/1539551

在 Exchange 运维的过程中，大家可能会遇到用户通过 OWA 成功重置完密码后，新旧密码可以同时登陆的问题，域计算机登录和 outlook 不会出现此问题。

这个问题是由于 IIS 的 TokenTTL 时间设置错误导致的，TokenTTL 是指令牌缓存的时间，是保证 IIS 性能的一项设置，默认为 15 分钟。

解决方法：

1 重启 Exchange 前端服务器 IIS 服务

2 通过注册表项限定，创建方法：

运行注册表编辑器（Regedt32.exe 或 Regedit.exe)。

从 HKEY_LOCAL_MACHIME 子树，请转到下面的项：

\System\CurrentControlSet\Services\InetInfo\Parameters

在编辑菜单上单击添加值，添加下列选项：

值名称: UserTokenTTL

数据类型: REG_DWORD

数据： （令牌缓存-30 秒的最小秒数）

需要注意的地方，根据 IIS 版本的不同最小值是有区别的。

IIS4.0 最小为 30 秒，IIS5.0 最小值为 1 秒。所以即便这个值设置成 0 为 0，系统依然使用以上最小值作为令牌缓存失效时间。

IIS6.0 以上版本最小值可以设置为 0。但是一个很关键的问题，当设置为 0 时，表示不清空令牌缓存，也就是说比如用户通过 OWA 更改完密码后旧密码将始终能够登录，直到你重启了 IIS 服务为止。

浪潮 NF5280M3 安装 Windows Server 2008 R2 注意事项

作者：王春海 来源：http://wangchunhai.blog.51cto.com/225186/1551345

近期某单位购买了几台浪潮英信 NF5280M3 服务器，每台服务器配置有 2 个 E5-2630 的 CPU、16GB 内存、5 块 2.5 寸的 600GB 的 SAS 硬盘。服务器到位之后，在安装 Windows Server 2008 R2 的时候出了问题，无论是使用 Windows Server 2008 R2 安装光盘引导安装（由于 Windows 2008 R2 安装光盘中不带 NF 5280M3 的驱动程序），还是使用浪潮睿捷系统安装，在完成第一步骤的操作之后，都不能引导。

我到现场之后，首先进入 RAID 卡配置界面，发现 RAID 配置没有问题，5 块硬盘做 RAID5，划分了两块逻辑磁盘，第 1 块 147GB，第 2 块 2TB。然后进入 CMOS 设置，检查 CMOS 设置，也没发现问题（将 UEFI 改为 Legacy 引导也不行），服务器仍然不能引导。

我用 Windows Server 2008 R2 光盘启动，随机带的 RAID 卡驱动没有经过 WHQL 认证，然后使用浪潮睿捷光盘安装成功，解决步骤记录如下。

一、驱动程序没有经过 WHQL 认证

NF 5280M3 使用的是一个“6605/6605 Q”的 RAID 卡，Windows Server 2008 R2 安装光盘中没有集成这个驱动程序，浪潮随机带的驱动光盘中虽然带了 Windows Server 2008 R2 及 Windows Server 2012 R2 的驱动程序，但这个驱动程序没有经过 Microsoft 的 WHQL 认证。对于 RAID 卡来说，如果没有经过 WHQL 认证，在安装操作系统的时候，是不能将系统安装在该磁盘上的。

（1）使用 Windows Server 2008 R2 安装光盘启动，没有找到硬盘，单击“加载驱动程序”，在“加载驱动程序”对话框，单击 “浏览”按钮。此时从光盘中取出 Windows Server 2008 R2 安装光盘，插入浪潮随机带的 RAID 卡驱动程序光盘，浏览选择驱动程序文件夹，如图 1-3 所示。

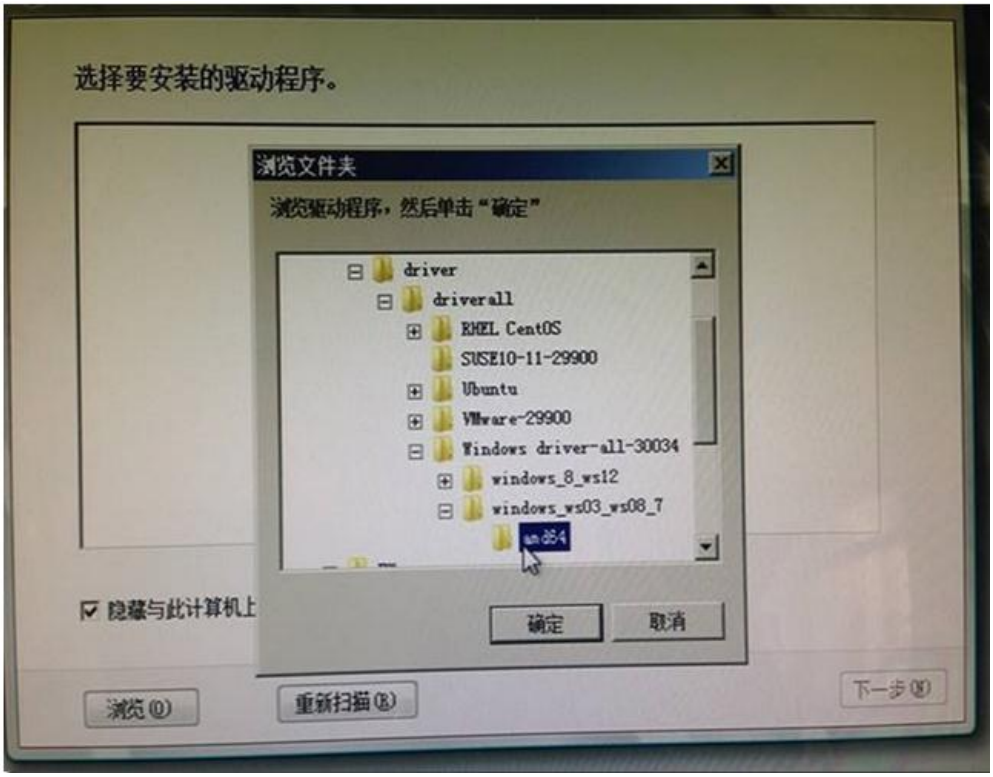


图 1-3 浏览驱动程序

(2) 浏览驱动程序位置后，可能会返回到安装界面，再重新加载一下驱动程序，才能出现“选择要安装的驱动程序”，单击“下一步”按钮，完成驱动程序的加载。

(3)在加载驱动程序后，可以“认出”磁盘，但此时安装程序提示，无法将 Windows 安装到这个磁盘，如图 1-5 所示。

图 1-5 无法将 Windows 安装到这个磁盘

在图 1-5 中，“磁盘 0”是 146.5GB，是 RAID 卡划分的第 1 个逻辑卷，这是我们所规划的。但是，如果再次用光盘启动，加载 RAID 卡驱动，则有时候“磁盘 0”则是 2048GB，是 RAID 卡划分的第 2 个逻辑卷，这与我们所规划的就有差别了。我想这可能是厂商提供的 RAID 卡驱动程序兼容性不好导致。另外，当出现图 1-5 的提示时，根据我的经验判断，可能是当前的 RAID 卡驱动程序没有经过 WQL 认证。

二、划分一个逻辑卷安装系统成功

既然正常用 Windows Server 2008 R2 光盘启动不能安装系统，此时考虑使用浪潮服务器带的“睿捷系统智能安装软件”安装 Windows Server 2008 R2，安装主要步骤：

(1) 光驱插入“睿捷系统智能安装软件”安装光盘，重新启动服务器，选择安装 Windows Server 2008 R2，在进行必要的配置之后，换入 Windows Server 2008 R2 安装光盘，向导复制 Windows Server 2008 R2 的安装文件，然后弹出光盘，重新启动计算机，“睿捷系统智能安装软件”向导界面如图 1-7 所示。

图 1-7 安装向导

（2）但按照向导安装完成之后，仍然不能启动。

后来我想，既然浪潮随机带了配置光盘，应该是可以安装系统的，但按照向导配置之后，不能启动，可能是我们的步骤与厂商的实验或测试环境不同。而唯一不同的，可能就是我们用 RAID 卡划分了两个逻辑卷，而厂商可能测试了一个逻辑卷。考虑到这之后，我重新用“睿捷系统智能安装软件”软件光盘启动，并且将 5 块硬盘划分为一个 RAID5 的卷，然后再复制 Windows Server 2008 R2 安装文件，成功完成安装。

(3）在安装的过程中，安装向导提示“Windows 无法验证此驱动程序软件的发布者”提示，也验证了原来的猜测—RAID 卡驱动程序没有经过 WHQL 验证。如图 1-8 所示，选择“始终安装此驱动程序软件”，以完成驱动程序的加载及操作系统的安装。

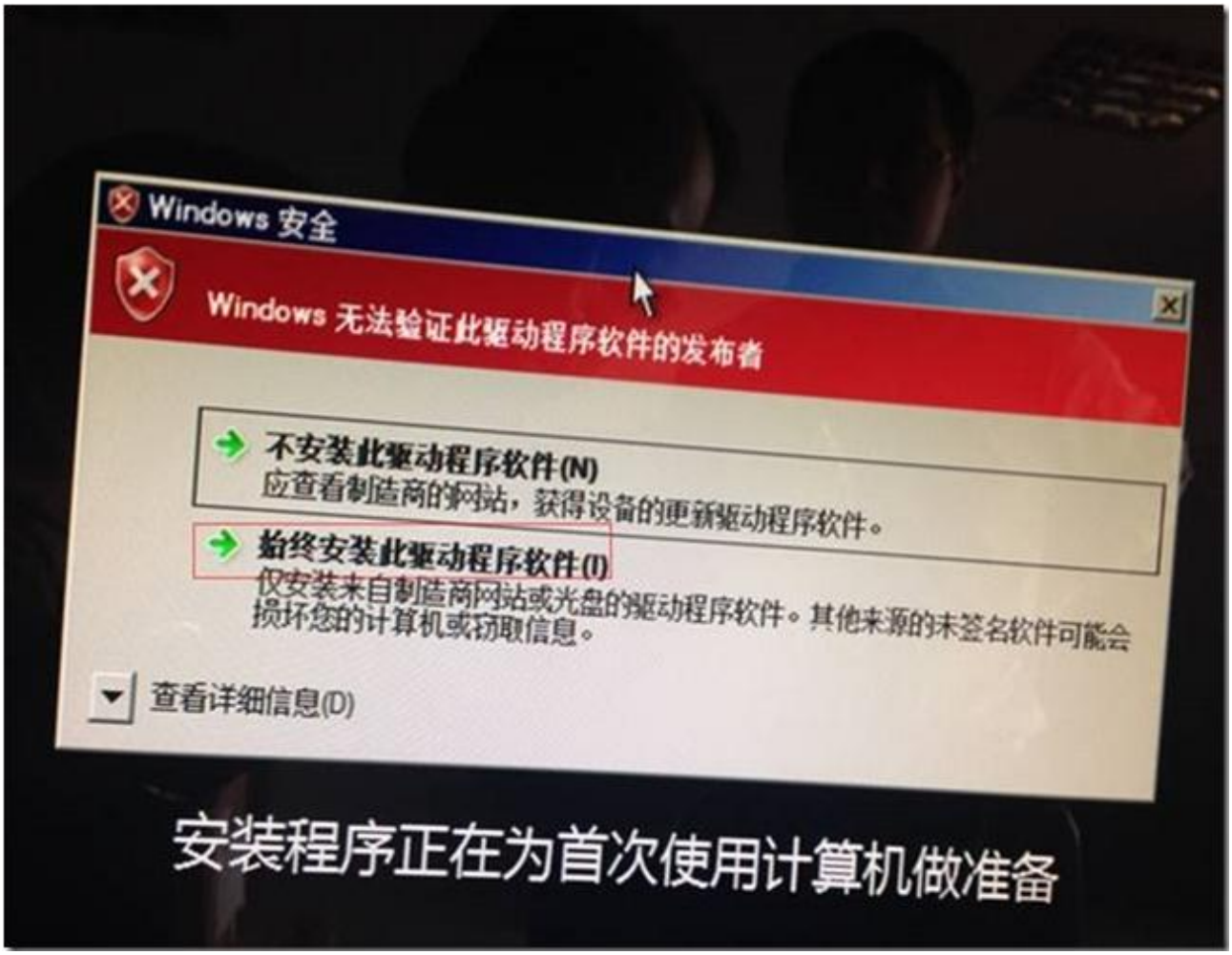


图 1-8 安装驱动程序

三、如果没有向导光盘怎么安装系统

如果你的服务器是自己配置的 RAID 卡，但你没有类似浪潮“睿捷系统智能安装软件”光盘，怎么安装 Windows 2008 或 2012？这时候我们可以采用“折中”的方法，你可以在你的服务器上配一块普通的硬盘，例如一块 300G 的 SATA 或一块 60 到 120GB 的固态硬盘，可以先将操作系统安装在这块独立的硬盘上，在安装好操作系统之后，再安装 RAID 卡的驱动程序，和图 1-6 一样，忽略 WHQL 签名，将 RAID 卡配置的磁盘做数据盘即可以使用。这是因为没有经过 WHQL 认证的驱动程序虽然在系统安装时不能忽略，但在系统安装完成之后是可以忽略使用的。

大数据时代的全能日志分析专家--Splunk 安装与实践

作者：李晨光 来源：http://chenguang.blog.51cto.com/350944/1548355

0.背 景

随着大家对网络安全意识的提高，企业网管理人员，必须对 IT 基础设置进行监控及安全事件的管理，管理数据的数量和种类非常巨大，那么就需要有一款能否分析各种日志数据的工具，经过长期实践，为大家推荐 Splunk 这么一款全能型分析工具。

1．Splunk 简介

Splunk 是一款功能强大的、记录详细的日志分析软件，Splunk 是基于原始日志数据（Raw data）内容建立索引，保存索引的同时也保存原始日志内容，在大数据时代，种类繁多的日志如何能快速分析找到你需要的内容呢，你需要一个更加方便智能的工具，那就是 Splunk。它能处理常规的日志格式，比如 Apache、Squid、系统日志、邮件日志等这些对所有日志先进行索引，然后可以交叉查询，支持复杂的查询语句，最后通过直观的方式表现出来。它与其他开源日志分析工具不同的是，操作界面支持全中文，而且对于中文版操作系统的日志收集非常不错,目前它的商业版本价格的确不便宜(国内天旦、精诚华厦微科都在代理这款产品，商务可联系他们)。下面我们先看看怎么安装和基本使用吧。

2. Splunk 索引数据内容

Splunk 的索引范围涵盖应用、服务器、网络设备中的所有日志、配置、信息、trap、告警、度量以及其他系统性能数据。可灵活地从文件、网络端口、数据库、自定义 API 和接口中实时或按需访问数据。

它的索引对原始数据的完整性无影响。

3．Splunk 安装

首先到官方 http://www.splunk.com/download 注册一个账号下载对应的操作系统版本（截止目前最新版本为 6.1.3），安装时记住关闭 SELinux 功能，另外注意一点，如果要通过 WMI 的方式来搜集 Windows（中文版）日志的话，那么 Splunk 建议装在 Windows 操作系统(须 4GB 以上可用空间)上。如果收集的日志主要是各种网络设备及 Linux 系统日志建议装在类 Unix 系统上。下面以 Redhat Linux 系统安装 Splunk 为例讲解安装过程，启动过程如图 1 所示。

（1）.安装软件包

#rpm –ivh splunk-4.1.7.95063-linux-2.6-x86_64.rpm

Splunk 安装路径在/opt/splunk,这个路径各种 UNIX/Linux 系统都一样。

（2）.关闭 Selinux

#setenforce 0

(3).启动 splunk，命令如下：

#/opt/splunk/bin/splunk start

(4).浏览 Splunk Web 接口，在浏览器中输入以下地址：

http://localhost.localdomain:8000

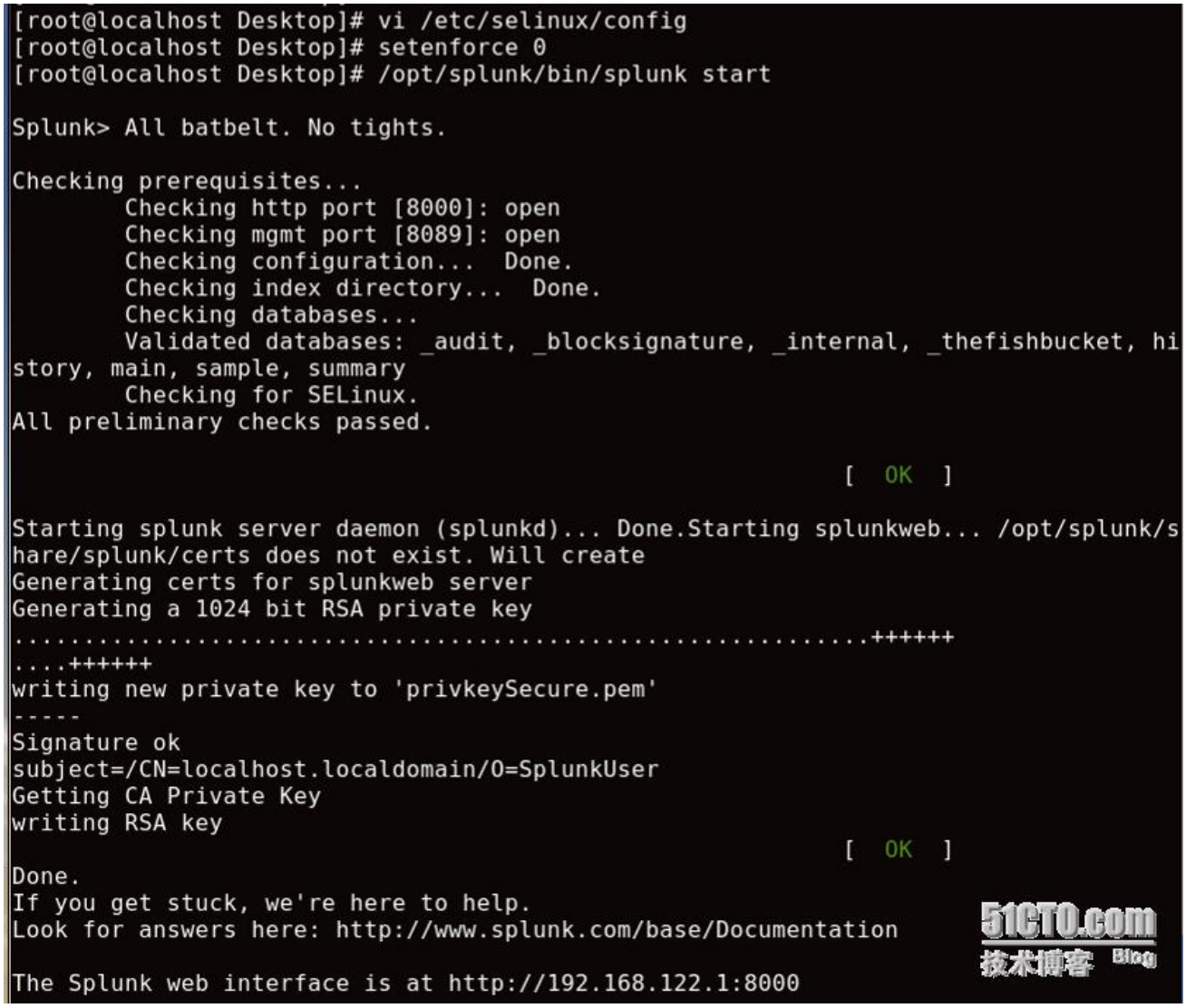


图 1 Linux 下安装 Splunk

用 netstat 命令检查 8000 端口是否处于监听状态。

#netstat -ant

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 0 0.0.0.0:8000 0.0.0.0:* LISTEN

tcp 0 0 0.0.0.0:8089 0.0.0.0:* LISTEN

我们看到以上信息输出即可通过网址访问。

4.设置自动运行

1）. 设置开机自动启动

#ln -s /usr/local/splunk /bin/splunk/etc/rc2.d/S80splunk

2）．设置到服务里面

#ln -s /usr/local/splunk /bin/splunk/etc/init.d/splunk

5．系统配置

下面我们通过配置来收集客户端的日志。

1）通过 Syslog 收集 Cisco 网络设备的日志

在 Cisco 网络设备上的配置命令一般为：

logging <syslog server IP Address>

logging trap <severity>

Splunk 默认使用 UDP 514 端口来监听 syslog 消息。例如：

logging 192.168.122.1

logging trap warning

2）通过 Syslog 收集 Linux 主机的日志

在 Linux 主机上的配置一般为修改/etc/syslog.conf 配置，添加以下两行：

Send syslog to Splunk server

*.<severity> @<syslog server IPAddress>

如：

Send syslog to Splunk server

*.debug @192.168.122.1

3）通过 WMI 来收集 Windows 主机的日志

首先要确保运行 Splunk 服务（在服务管理器中显示为 Splunkd）的帐号有权限读取远程 Windows 机器的 WMI 信息。在《Unix/Linux 日志分析和流量监控》一书中的第 14 章中还会讲到利用 WMI 收集 Windows 日志。

然后，在 Splunk 服务器上做一下简单的配置。这里假设 Splunk 的安装路径默认为 C:\Program Files\Splunk。在 C:\Program Files\Splunk\etc\system\local 文件下修改 inputs.conf 文件，添加以下内容：

```
[script://$SPLUNK_HOME\bin\scripts\splunk-wmi.py]
interval = 10
source = wmi
sourcetype = wmi
disabled = 0
```

接着，在同一目录中新建一个文本文件，命名为 wmi.conf，并添加以下内容：

```
[WMI:<Name>]
server = <Remote Windows Host IPAddress>
interval = 60
event_log_file = <Event log Type>
disabled = 0
```

比如监控 IP 地址为 192.168.122.10 的 Windows 主机上 Application 和 System 的 Event Log：

```
[WMI:AppAndSys]
server = 192.168.122.1
interval = 60
event_log_file = Application, System
disabled = 0
```

其实还可以通过 Syslog 来收集 Windows 的日志，这里可以用一个免费工具 NTSyslog(下载)。

6.设置日志分析目录

当首次进入 Web 界面后，需要重设密码并添加数据。进入系统可以将默认语言选择为中文，开始导入数据，如图 2 所示。

图 2 导入数据

选择数据源（从本地），接着选“从文件和目录”，选择/var/log 即可。如图 3 所示。从图中我们也可以看出 Splunk 默认支持的日志种类很多，包含大多数运维人员平时工作中所需要分析的日志类型。

图 3 选择本地数据源

点击应用菜单下方的 search 即可看到生成的日志报告(比如 cron 日志，mail 日志。当然也可以把我们所需要记录的日志比如 php 错误日志等都输出到/var/log 目录下，对其进行分析)。

7 .Splunk 搜索的使用

系统中的搜索工具栏是 Splunk 最强大的工具，为了学习 Splunk，我们先在 http://www.splunk.com/base/images/Tutorial/Sampledata.zip 下载一个演示文件。我们学习如何添加数据，首先向 Splunk 添加示例数据方法如下：

在 Splunk 首页中点击右上角的 Home 按钮，再选择添加数据，选择服务器本地文件，当你选择正确系统提示：“Use auto-detected source type:access_combined_wcookie”最后保存配置，当系统提示索引建立后就可以查看日志。

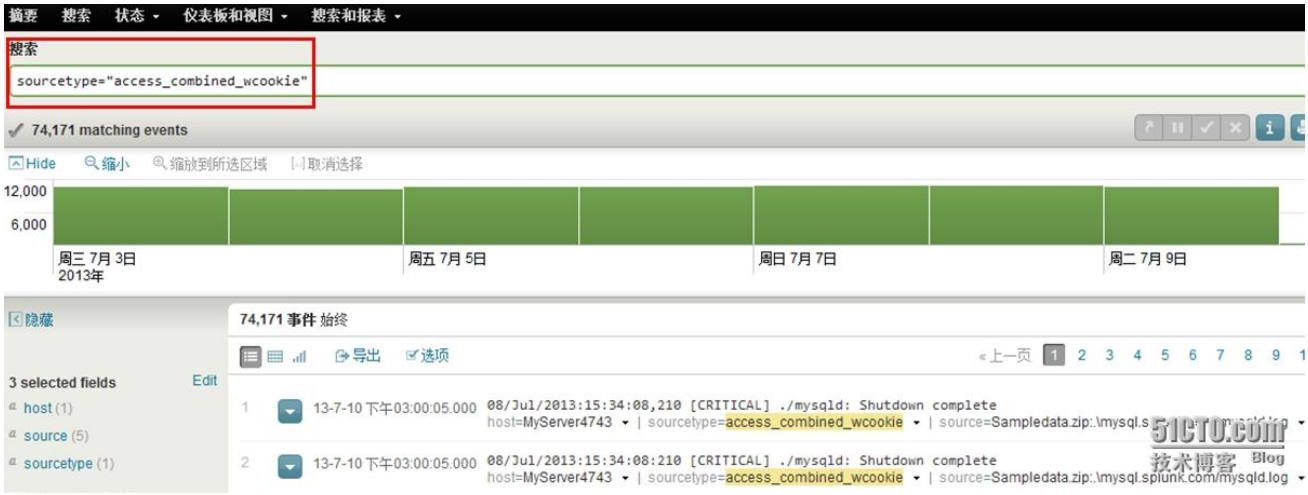


图 4 开始搜索

我们看看仪表盘的内容，读者应该已经熟悉搜索栏及时间范围选择，摘要仪表板上也有这些内容。但搜索仪表板上还包含其他内容，比如事件记录、时间轴、字段菜单及检索到的事件列表或搜索结果。

1) .匹配及扫描事件记录：在搜索中，Splunk 在检索时将显示两组事件记录：一组为匹配事件记录，另一组为已扫描事件记录。搜索完成后，时间轴上方的记录显示的是匹配事件的总数。时间轴下方事件列表上方的记录显示用户所选时间范围内的时间数目。稍后可以看到，当向下分析事件时，此数目会发生变化。

2) .事件的时间轴：时间轴能直观的显示出每一时刻发生的事件。当时间轴随着搜索结果不断更新时，可能会注意到有条状图案。每一条状图案的高度表示时间记录。时间轴的峰值和谷值可表示活动高峰期或服务器停机。此时，时间轴可有效用于强调时间模式或调查各事件活动的高峰期和低谷期。时间轴选项位于时间轴上方。还可以放大或缩小图表。

3）.字段菜单：前面说过将数据编入索引时，Splunk 可自动按名称和值的格式识别并生成数据信息，我们把这称作是字段。当您进行搜索时，Splunk 将把其从字段菜单上识别的所有字段列在搜索结果旁边。您可以选择其他字段来显示您搜索的事件。所选字段都已被设置为搜索结果可见格式。将默认显示主机、源及源类型。其它字段是 Splunk 从您的搜索结果中抽取的。

4）.事件查看器：事件查看器将显示 Splunk 搜索到的与您的搜索匹配的事件。事件查看器位于时间轴下方。事件默认显示为列表，您也可以选用表格查看。选择按表格形式查看事件时，表格只显示已选字段。

8.Splunk 搜索实例解析

我们先构造一个场景，假如有人投诉网站，说在提交表单时总是提示有某个 IP 地址错误， 10.2.1.44，这时我们该如何利用搜索功能来查找问题呢？

我们可以输入如下内容：sourcetype=access_combined_wcookie10.2.1.44

当然，如果你不知道数据源，那么你也可以直接输入 IP 地址，这样匹配的条目会非常多，如果能精确找到数据源就很容易找到问题。

图 5　使用搜索功能

access_combined_wcookie 代表数据源，要根据你提交的日志而定。注意，当您在搜索栏中输入的同时，将弹出 Splunk“搜索助手”这个很重要，可以帮助你解决搜索中的很多问题。就如同 MS Office 中的帮助一样好用。为了缩小范围我们做如下操作，我们还应该在搜索栏中键入 purchase：

sourcetype=access_combined_wcookie10.2.1.44 purchase

图 6　缩小搜索范围

请看左上角搜到的日志从 109 降到 83 条。注意，搜索关键词时，不用区分大小写。

9.使用布尔运算符查找日志

Apache 服务器日志中发现大部分事件的状态码为“200”，它代表“成功”。现在有人投诉网站出现了问题，那么就要找出不是 200 的日志。我们使用布尔运算方法。

图 7　使用布尔运算符查找

这时匹配条数骤减到 31 条。这时发现了 HTTP 服务器（503）错误，用这个方法可以快速排除无关事件。使用布尔运算符可进行搜索的信息更多，Splunk 支持的布尔运算符有与、或和非所以第四步的搜索和下述语句相同：

sourcetype=access_* AND 10.2.1.44 ANDpurchase NOT 200

当搜索中含有布尔表达式时，运算符须全部大写。使用括号将有关表达式组合起来，以便进行更复杂的搜索。计算布尔表达式时，Splunk 将从最里面的括号开始运算，接着运算括号外面的下一个值对。当括号内的所有运算符都运行完成，Splunk 将先计算或子句，然后计算和或者非子句。

10.使用时间轴功能

现在您已经确认存在问题类型，现在您想找到导致问题的原因。从发现顾客无法购买的那次搜索开始，继续进行下面的步骤。时间轴上的各柱状体，代表搜索的匹配事件发生的时间。滑动鼠标，选中其中一个柱状体，将弹出工具提示，并显示时间数目和该柱距的原始时间戳，1 个柱状体=1 分钟，这个单位根据你的选择而动态变化，这样您的搜索将仅限于您所选定的 1 小时内所发生的事件，如图 8 所示。

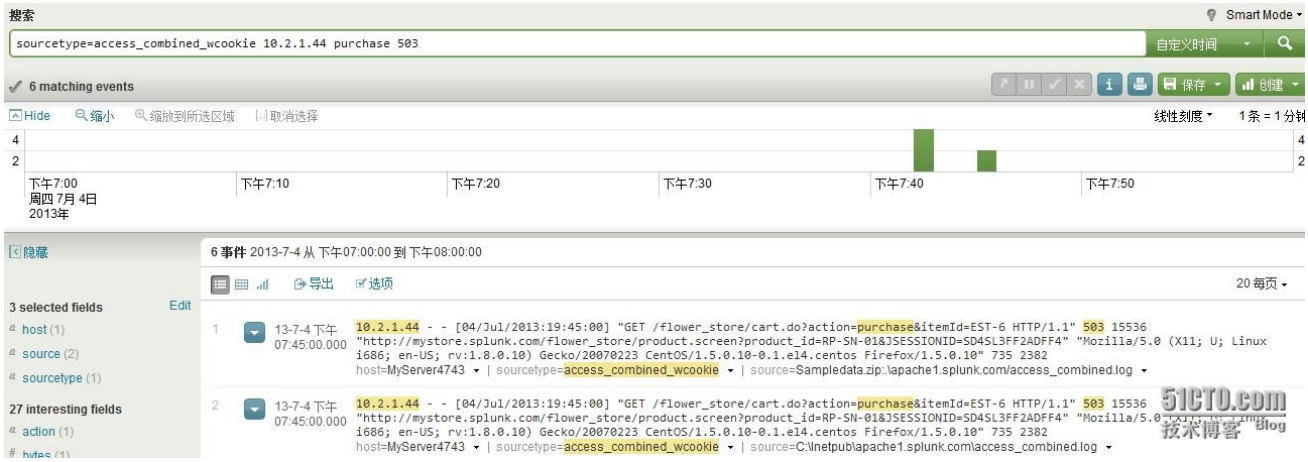


图 8 使用时间轴

Splunk 支持使用星号（*）通配符来搜索“所有”或根据关键词的部分进行模糊检索事件。该搜索可告诉 Splunk 希望看到在这段时间内发生的所有事件。

时间轴的其他功能：

- 点击选择上述的所有时间轴，可再次显示所有时间；
- 点击放大，可锁定与您的搜索匹配的选定事件范围；
- 点击缩小，可扩展时间轴，看到更多事件；

11．故障定位方面的应用

Splunk 能通过搜索出日志中的重要关键字来挖掘出网络设备日志中最有价值的信息。搜索关键字“up OR down”查看日志中存在接口连接情况，splunk 将信息转换成时间分布图，使我们更快捷地查看当天或者过去几天设备接口连接状态。

搜索关键字“duplicate”，发现有少量存在 IP 地址冲突的地址，其中地址冲突所发生的时间以及冲突的源主机 MAC 地址都可以一目了然；搜索关键字“SYNflood”，可在防火墙日志中查找 SYN 攻击事件；搜索关键字“power”可快速查找重要设备是否会出现“poweroff”的情况。

搜索关键字“deny”可查找核心交换机上丢弃数据包的具体情况，根据这些情况可以统计一些经常出现的被丢弃数据包源头。输入 EventCode=6005 or EventCode=6006 查询可以掌握机器的开关机情况，主要是提取 6006 的事件和 6005 的事件信息系统，思路是在 windows 中打开 eventvwr.msc（事件查看器）程序打开事件查看器，在左侧窗口中选择“系统”，从右侧系统事件中查找事件 ID 为 6005、6006 的事件（事件 ID 号为 6005 的事件表示事件日志服务已启动，即开机事件 ID：6006 表示关机），它们对应的时间就分别是开机时间和关机时间。

注意：Windows 事件 ID 含义详情请点击这里

12.看视频学用 Splunk 分析日志

最后轻松一下，大家打开这个链接（前提是你的浏览器支持 flash）观看 Splunk 视频指南： http://www.tudou.com/programs/view/7iXM5WfXpDg/

技巧：

在试用版的 Splunk 中有 500MB 日志的限制（个人还无法突破限制），如果你直接将主机架设在生产环境，很快就到达上限，也许你一着急，就把他卸载了，从而错失真么一款优秀的工具，建议开始测试时，找几个典型测试设备在实验室进行功能测试，经过自评价后，有必要在联系商务人员，他们可以免费为您企业进行安装调试。

另外，如果你懒得注册账户，想直接下载 Splunk 4.1.7 请点击这里：

1) Windows 平台 64 位 下载地址

2) RedHat Linux 平台 2.6 内核 64 位 下载地址

3) Solaris 9/10 (64 bit) 下载地址

4）更多 Splunk 学习手册

两个局域网安全互通方案 2：by GRE and linux server&深入理解 GRE

作者：hh2o 来源：http://h2ofly.blog.51cto.com/6834926/1544860

【第一、需求描述】

办公网和 IDC 两个局域网(or，任意两个不同局域网)，相互隔离。但是在日常运维、研发过程中，需要在办公网访问 IDC 网络。如果都通过公网 ip 绕，既不方便，也不安全。如果拉专线，是最稳定可靠的办法。但是作为技术屌丝，想为公司省点钱（这也可以看作是技术价值的一部分），所以打算使用其他方案（当然是免费的方案，或者这样说，使用已有资源的方案。服务器当然也需要花钱的，但是你可以使用已有服务器来完成这个方案）解决这个需求。

【第二、背景介绍】

办公网有 lan192.168.1.0/24，并通过固定的公网 IP 上网；

IDC 有 lan10.1.1.0/24，并有多个公网 ip，两者通过公网 ip 互联

这就遇到问题：办公网怎样能直接访问 IDC 的内网，至少普通用户看起来是可以直接访问机房的内网的。你让每个用户都拨 VPN？很悲催，还不好管理。

这里有介绍两个方案：

1、 点到端的 VPN 方案（架设成功之后，你可以设置为端到端的方案），之前写过一篇文章，具体可以看《创业公司办公网络安全稳定访问机房网络方案 1:byVPN》

http://h2ofly.blog.51cto.com/6834926/1529888

2、GRE 方案。如果公司有多余的固定的公网 ip 或者路由器本身支持 GRE，建议使用本方案。why？不解释，哈哈。你可以对比、再使用两个方案之后就有感觉了。

【第三、方案实施】

说了这么多，实际操作下印象就更深了

办公网路由器（linux 服务器实现）：局域网 ip：192.168.1.254，公网 ip180.1.1.1 配置

cat /usr/local/admin/gre.sh#并把改脚本加入开机启动

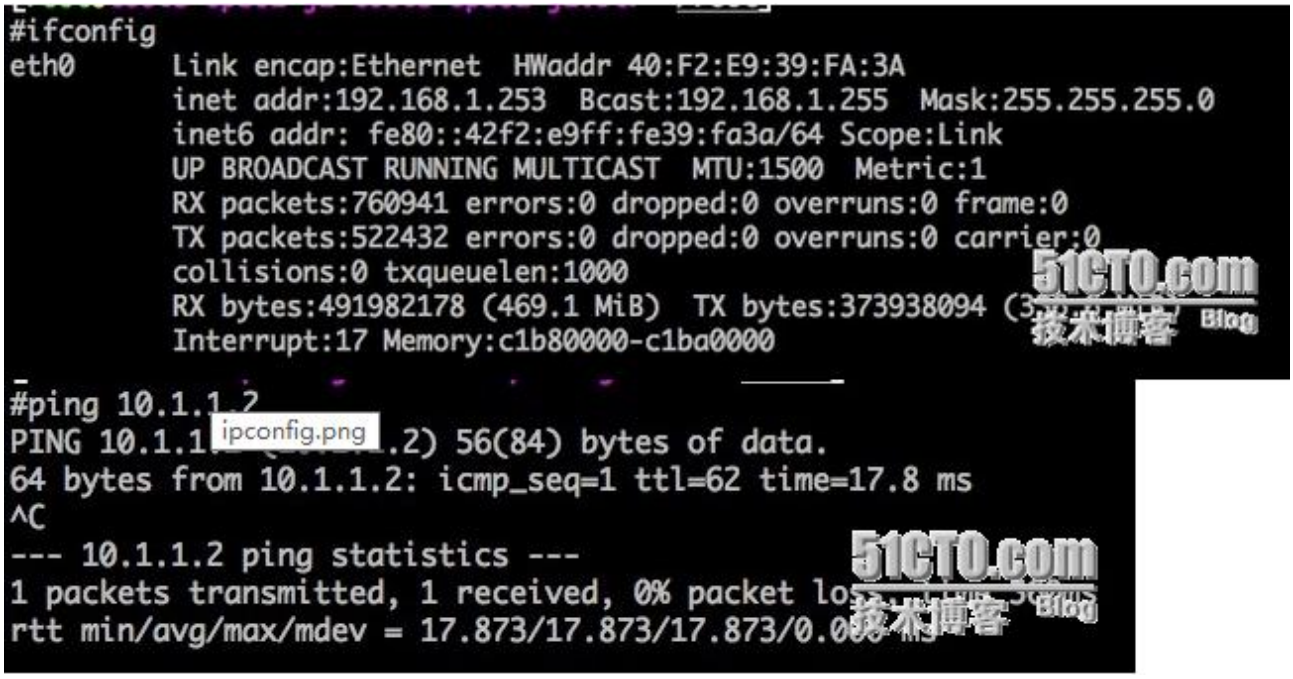
```
#!/bin/bash
modprobe ip_gre#加载 gre 模块
ip tunnel add office mode gre remote 110.2.2.2 local 180.1.1.1 ttl 255#建立 tunnel 名字叫 office 的 device（可自定义），使用 gre mode。指定远端的 ip 是 110.2.2.2，本地 ip 是 180.1.1.1。这里为了提升安全性，你可以配置 iptables，公网 ip 只接收来自 110.2.2.2 的包，其他的都 drop 掉。
ip link set office up#启动 device office
ip link set office up mtu 1500#设置 mtu 为 1500
ip addr add 192.192.192.2/24 dev office #为 office 添加 ip192.192.192.2
echo 1 > /proc/sys/net/ipv4/ip_forward #让服务器支持转发
ip route add 10.1.1.0/24 dev office #添加路由，含义是：到 10.1.1.0/24 的包，由 office 设备负责转发
iptables -t nat -A POSTROUTING -d 10.1.1.0/24 -j SNAT --to 192.192.192.2#否则 192.168.1.x 等机器访问 10.1.1.x 网段不通
```

IDC 路由器（linux 服务器实现）：局域网 ip：10.1.1.1，公网 ip110.2.2.2 配置 cat /usr/local/admin/gre.sh#并把改脚本加入开机启动

```
#!/bin/bash
modprobe ip_gre
ip tunnel add office mode gre remote 180.1.1.1 local 110.2.2.2 ttl 255
ip link set office up
ip link set office up mtu 1500
ip addr add 192.192.192.1/24 dev office#为 office 添加 ip192.192.192.1
echo 1 > /proc/sys/net/ipv4/ip_forward
ip route add 192.168.1.0/24 dev office
iptables -t nat -A POSTROUTING -s 192.192.192.2 -d 10.1.0.0/16 -j SNAT --to 10.1.1.1#否则 192.168.1.X 等机器访问 10.1.1.x 网段不通
iptables -A FORWARD -s 192.192.192.2 -m state --state NEW -m tcp -p tcp --dport 3306 -j DROP #禁止直接访问线上的 3306，防止内网被破
```

注：为了预防偶然因数 gre turnel 断掉，你可以自己研究监控脚本。在检测到网络不同的时候，再运行这个脚本（运行之间请注意 device office 是否已存在等），这里就不详细介绍了。请自己动手。

【第四、测试效果】



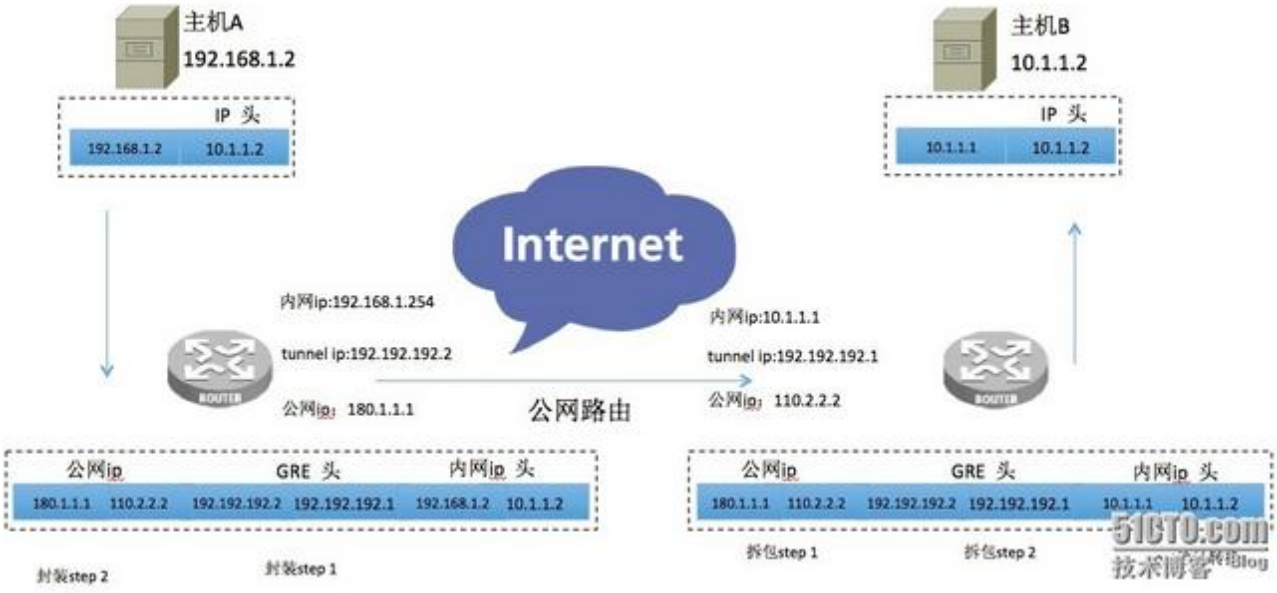
【第五、深入理解 GRE】

操作之后，需要对背景和理论更加熟悉才能在遇到异常的时候才能处理的游刃有余。

GRE（通用路由协议封装）协议，由 Cisco 和 Net-smiths 等公司于 1994 年提交给 IETF，它对部分网络层协议（ip）的数据报进行封装。它规定了如何用一种网络协议去封装另一种网络协议，很多网络设备都支持该协议。说得直白点，就是 gre 将普通的包（如 ip 包）封装了，又安装普通的 ip 包的路由方式进行路由，相当于 ip 包外面再封装一层 gre 包。在本例子中，在 gre 包外围实际上又有一层公网的 ip 包。

GRE 使用 tunnel（隧道）技术，数据报在 tunnel 的两端封装，并在这个通路上传输，到另外一端的时候解封装。你可以认为 tunnel 是一个虚拟的点对点的连接。（实际 PointToPoint 连接之后，加上路由协议及 nat 技术，就可以把两个隔绝的局域网连接在一起，就实现了 NetToNet 的互联。）

一般 gre turnel 是在多个网络设备（一般为路由器）之间建立。因为 linux 服务具备路由转发功能，所以当您缺少专业的路由设备的时候，可使用 linux 服务器实现 gre turnel 的建立，这也说明 linux 的强大啊（实际上巨大多数网络设备都使用 unix 或 linux 系统）



我对 gre 的理解如下：

（0）gre 的 turnel 的打通

1、 这个过程就是双方建立 turnel 的过程。

（1）局域网路由过

1、主机 A 发送一个源为 192.168.1.2，目的为 10.1.1.2 的包

（2）封装过程

1、根据内网路由，可能是你的默认路由网关将之路由至 192.168.1.254

2、192.168.1.254 第一次封装包，增加增加 gre 包头，说明包的目的地址 192.192.192.1 和源地址 192.192.192.2。

3、192.168.1.254 第 2 次封装包，增加公网的包头（否则在公网上无法路由），说明包的目的地址 110.2.2.2 和源地址 180.1.1.1。

4、192.168.1.254 把所有到 10.1.1.0/24 的包，都地址转换为从 192.192.192.2 出（snat）

（3）公网路由过程

1、经过 n 个路由设备，该包最终路由到 110.2.2.2

（4）拆包过程

1、B 端的路由器检测到是到达自己的 ip，就开始拆包

2、拆包之后发现有 GRE 协议，就进一步拆包

3、拆包之后发现目的地不是自己的内网 ip、发现自己本地做了 snat，就将至源 ip 替换为 10.1.1.1

（5）局域网路由

1、实际上从 10.1.1.1 出发的，到达目的地为 10.1.1.2 的包，无需路由，直接在局域网内广播。10.1.1.2 的机器确定是发送给自己的包，就接收。然后进一步处理了。

千万别手欠执行 stop slave

作者：贺春旸 来源：http://hcymysql.blog.51cto.com/5223301/1557704

今天我一个朋友，执行了 stop slave，给卡死了，结果 kill 进程 ID 也杀不死。

这是由于在主库上执行了一条很耗时的大 SQL，通过主从复制在从库接收过来后，SQL_THREAD 开始执行，这时你只要执行了 stop slave，立马就卡死，之后你再执行 show slave status\G;也会被卡住，必须等待那条大 SQL 执行完，才会结束 stop slave，除非你 pkill -9 mysql 进程。

下面就来重现一下，主库上执行全表更新 update sbtest set c='mariadb';等执行完以后，会记录到 binlog 日志里，然后在从库执行的时候，stop slave，就会卡住。

```
MariaDB [(none)]> show processlist;
+----+-----------------+-----------+------+---------+--------+---------------------------------+-------------------------------+---
--------+
| Id | User            | Host      | db   | Command | Time   | State                           | Info                          | Pr
ogress |
+----+-----------------+-----------+------+---------+--------+---------------------------------+-------------------------------+---
--------+
|  2 | event_scheduler | localhost | NULL | Daemon  |    948 | Waiting on empty queue          | NULL                          |
 0.000 |
|  8 | system user     |           | NULL | Connect |    754 | Waiting for master to send event | NULL                          |
 0.000 |
|  9 | system user     |           | test | Connect | 754525 | updating                        | update sbtest set c='mariadb' |
 0.000 |
| 11 | root            | localhost | NULL | Query   |     92 | Killing slave                   | stop slave                    |
 0.000 |
| 12 | root            | localhost | NULL | Query   |      0 | init                            | show processlist              |
 0.000 |
| 13 | root            | localhost | NULL | Query   |     32 | init                            | show slave status             |
 0.000 |
+----+-----------------+-----------+------+---------+--------+---------------------------------+-------------------------------+---
--------+
6 rows in set (0.00 sec)
```

```
MariaDB [(none)]> stop slave;
Query OK, 0 rows affected (3 min 48.34 sec)
```

最后，提醒一下，在执行 stop slave 的时候，一定要看下主库上的慢 SQL，避免出现被卡住的情况发生。

.NET 程序员项目开发必知必会—Dev 环境中的集成测试用例执行时上下文环境检查（实战）

> 作者：王清培 来源：http://wangqingpei557.blog.51cto.com/1009349/1553012

Microsoft.NET 解决方案，项目开发必知必会。

从这篇文章开始我将分享一系列我认为在实际工作中很有必要的一些.NET 项目开发的核心技术点，所以我称为必知必会。尽管这一些列是使用.NET/C#来展现，但是同样适用于其他类似的 OO 技术平台，这些技术点可能称不上完整的技术，但是它是经验的总结，是掉过多少坑之后的觉醒，所以有必要花几分钟时间记住它，在真实的项目开发中你就知道是多么的有帮助。好了，废话不说了，进入主题。

我们在开发服务时为了调试方便会在本地进行一个基本的模块测试，你也可以认为是集成测试，只不过你的测试用例不会覆盖到 80%以上，而是一些我们认为在开发时不是很放心的点才会编写适当的用例来测试它。

集成测试用例通常有多个执行上下文，对于我们开发人员来说我们的执行上下文通常都在本地，测试人员的上下文在测试环境中。开发人员的测试用来是不能够连接到其他环境中去的（当然视具体情况而定，有些用例很危险是不能够乱连接的，本文会讲如何解决），开发人员运行的集成测试用例所要访问的所有资源、服务都是在开发环境中的。这里依然存在但是，但是为了调试方便，我们还是需要能够在必要的时候连接到其他环境中去调试问题，为了能够真实的模拟出问题的环境、可真实的数据，我们需要能有一个这样的机制，在需要的时候我能够打开某个设置让其能够切换集成测试运行的环境上下文，其实说白了就是你所要连接的环境、数据源的连接地址。

本篇文章我们将通过一个简单的实例来了解如何简单的处理这中情况，这其实基于对测试用来不断重构后的效果。

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace OrderManager.Test
{
using ProductService.Contract;

/// <summary>
/// Product service integration tests.
/// </summary>
[TestClass]
public class ProductServiceIntegrationTest
{
    /// <summary>
    /// service address.
    /// </summary>
    public const string ServiceAddress = "http://dev.service.ProductService/";
```

```
    /// <summary>
    /// Product service get product by pid test.
    /// </summary>
    [TestMethod]
    public void ProductService_GetProductByPid_Test()
    {
        var serviceInstance = ProductServiceClient.CreateClient(ServiceAddress);
        var testResult = serviceInstance.GetProductByPid(0393844);

        Assert.AreNotEqual(testResult, null);
        Assert.AreEqual(testResult.Pid, 0393844);
    }
}
```

这是一个实际的集成测试用例代码，有一个当前测试类共用的服务地址，这个地址是 DEV 环境的，当然你也可以定义其他几个环境的服务地址，前提是环境是允许你连接的，那才有实际意义。
我们来看测试用例，它是一个查询方法测试用例，用来对 ProductServiceClient.GetProductByPid 服务方法进行测试，由于面向查询的操作是等幂的，不论我们查询多少次这个 ID 的 Product，都不会对数据造成影响，但是如果我们测试的是一个更新或者删除就会带来问题。

在 DEV 环境中，测试更新、删除用例没有问题，但是如果你的机器是能够连接到远程某个生产或者 PRD 测试上时会带来一定的危险性，特别是在忙的时候，加班加点的干进度，你很难记住你当前的机器的 host 配置中是否还连接着远程的生产机器上，或者根本就不需要配置 host 就能够连接到某个你不应该连接的环境上。

这是目前的问题，那么我们如何解决这个问题呢 ，我们通过对测试代码进行一个简单的重构就可以避免由于连接到不该连接的环境中运行危险的测试用例。

其实很多时候，重构真的能够帮助我们找到出口，就好比俗话说的："出口就在转角处 "，只有不断重构才能够逐渐的保证项目的质量，而这种效果是很难得的。

提取抽象基类，对测试要访问的环境进行明确的定义。

```
namespace OrderManager.Test
{
public abstract class ProductServiceIntegrationBase
{
    /// <summary>
    /// service address.
    /// </summary>
    protected const string ServiceAddressForDev = "http://dev.service.ProductService/";

    /// <summary>
```

```
    /// service address.
    /// </summary>
    protected const string ServiceAddressForPrd = "http://Prd.service.ProductService/";

    /// <summary>
    /// service address.
    /// </summary>
    protected const string ServiceAddressTest = "http://Test.service.ProductService/";
}
}
```

对具体的测试类消除重复代码，加入统一的构造方法。

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace OrderManager.Test
{
using ProductService.Contract;

/// <summary>
/// Product service integration tests.
/// </summary>
[TestClass]
public class ProductServiceIntegrationTest : ProductServiceIntegrationBase
{
    /// <summary>
    /// product service client.
    /// </summary>
    private ProductServiceClient serviceInstance;

    /// <summary>
    /// Initialization test instance.
    /// </summary>
    [TestInitialize]
    public void InitTestInstance()
    {
        serviceInstance = ProductServiceClient.CreateClient(ServiceAddressForDev/*for dev*/);
    }

    /// <summary>
    /// Product service get product by pid test.
    /// </summary>
    [TestMethod]
    public void ProductService_GetProductByPid_Test()
```

```
    {
        var testResult = serviceInstance.GetProductByPid(0393844);

        Assert.AreNotEqual(testResult, null);
        Assert.AreEqual(testResult.Pid, 0393844);
    }

    /// <summary>
    /// Product service delete search index test.
    /// </summary>
    [TestMethod]
    public void ProductService_DeleteProductSearchIndex_Test()
    {
        var testResult = serviceInstance.DeleteProductSearchIndex();

        Assert.IsTrue(testResult);
    }
}
```

消除重复代码后，我们需要加入对具体测试用例检查是否能够连接到某个环境中去。我加入了一个 DeleteProductSearchIndex 测试用例，该用例是用来测试删除搜索索引的，这个测试用例只能够在本地 DEV 环境中运行（你可能觉得这个删除接口不应该放在这个服务里，这里只是举一个例子，无需纠结）。

为了能够有一个检查机制能提醒开发人员你目前连接的地址是哪一个，我们需要借助于测试上下文。

重构后，我们看一下现在的测试代码结构。

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace OrderManager.Test
{
using ProductService.Contract;

/// <summary>
/// Product service integration tests.
/// </summary>
[TestClass]
public class ProductServiceIntegrationTest : ProductServiceIntegrationBase
{
    /// <summary>
    /// product service client.
    /// </summary>
    private ProductServiceClient serviceInstance;

    /// <summary>
```

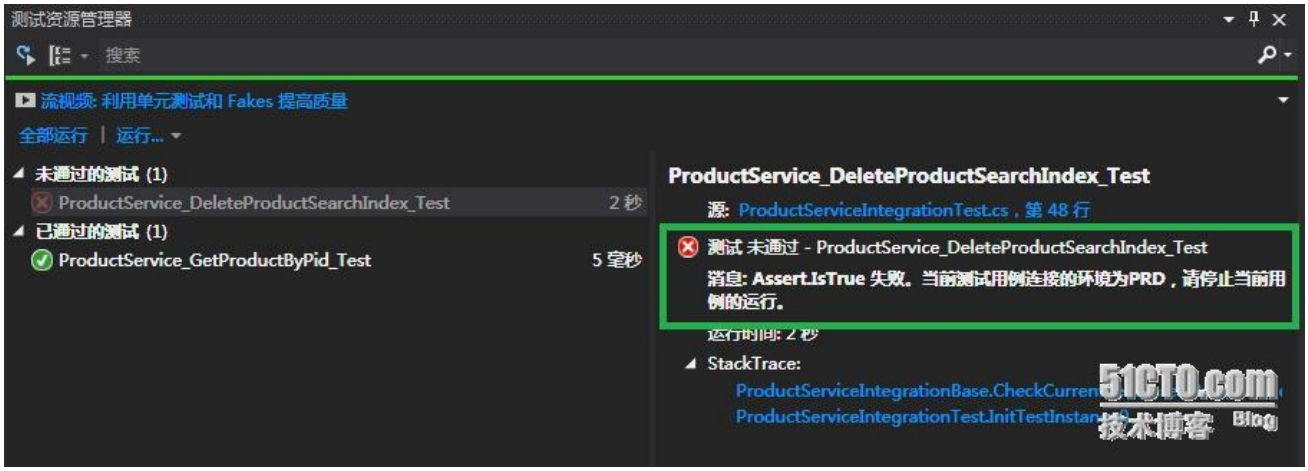
```
    /// Initialization test instance.
    /// </summary>
    [TestInitialize]
    public void InitTestInstance()
    {
        serviceInstance = ProductServiceClient.CreateClient(ServiceAddressForPrd/*for dev*/);

        this.CheckCurrentTestCaseIsRun(this.serviceInstance);//check current test case .
    }

    /// <summary>
    /// Product service get product by pid test.
    /// </summary>
    [TestMethod]
    public void ProductService_GetProductByPid_Test()
    {
        var testResult = serviceInstance.GetProductByPid(0393844);

        Assert.AreNotEqual(testResult, null);
        Assert.AreEqual(testResult.Pid, 0393844);
    }

    /// <summary>
    /// Product service delete search index test.
    /// </summary>
    [TestMethod]
    public void ProductService_DeleteProductSearchIndex_Test()
    {
        var testResult = serviceInstance.DeleteProductSearchIndex();

        Assert.IsTrue(testResult);
    }
}
}
```

我们加入了一个很重要的测试实例运行时方法 InitTestInstance，该方法会在测试用例每次实例化时先执行，在方法内部有一个用来检查当前测试用例运行的环境 this.CheckCurrentTestCaseIsRun(this.serviceInstance);//check current test case .，我们转到基类中。

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace OrderManager.Test
{
```

```
public abstract class ProductServiceIntegrationBase
{
    /// <summary>
    /// service address.
    /// </summary>
    protected const string ServiceAddressForDev = "http://dev.service.ProductService/";

    /// <summary>
    /// get service address.
    /// </summary>
    protected const string ServiceAddressForPrd = "http://Prd.service.ProductService/";

    /// <summary>
    /// service address.
    /// </summary>
    protected const string ServiceAddressTest = "http://Test.service.ProductService/";

    /// <summary>
    /// Test context .
    /// </summary>
    public TestContext TestContext { get; set; }

    /// <summary>
    /// is check is run for current test case.
    /// </summary>
    protected void CheckCurrentTestCaseIsRun(ProductService.Contract.ProductServiceClient
testObject)
    {
        if (testObject.ServiceAddress.Equals(ServiceAddressForPrd))// Prd 环境，需要小心检查
        {
            if (this.TestContext.TestName.Equals("ProductService_DeleteProductSearchIndex_Test"))
                Assert.IsTrue(false, "当前测试用例连接的环境为 PRD，请停止当前用例的运行。");
        }
        else if (testObject.ServiceAddress.Equals(ServiceAddressTest))//Test 环境，检查约定几个用例
        {
            if (this.TestContext.TestName.Equals("ProductService_DeleteProductSearchIndex_Test"))
                Assert.IsTrue(false, "当前测试用例连接的环境为 TEST，为了不破坏 TEST 环境，请停止用例的运
行。");
        }
    }
}
```

在检查方法中我们使用简单的判断某个用例不能够在 PRD、TEST 环境下执行，虽然判断有点简单，但是在真实的项目中足够了，简单有时候是一种设计思想。我们运行所有的测试用例，查看各个状态。



一目了然，更为重要的是它不会影响你对其他用例的执行。当你在深夜 12 点排查问题的时候，你很难控制自己的眼花、体虚导致的用例执行错误带来的大问题，甚至是无法挽回的的错误。

此文献给那些跟我一样的.NET 程序员们，通过简单的重构，我们放开了自己。

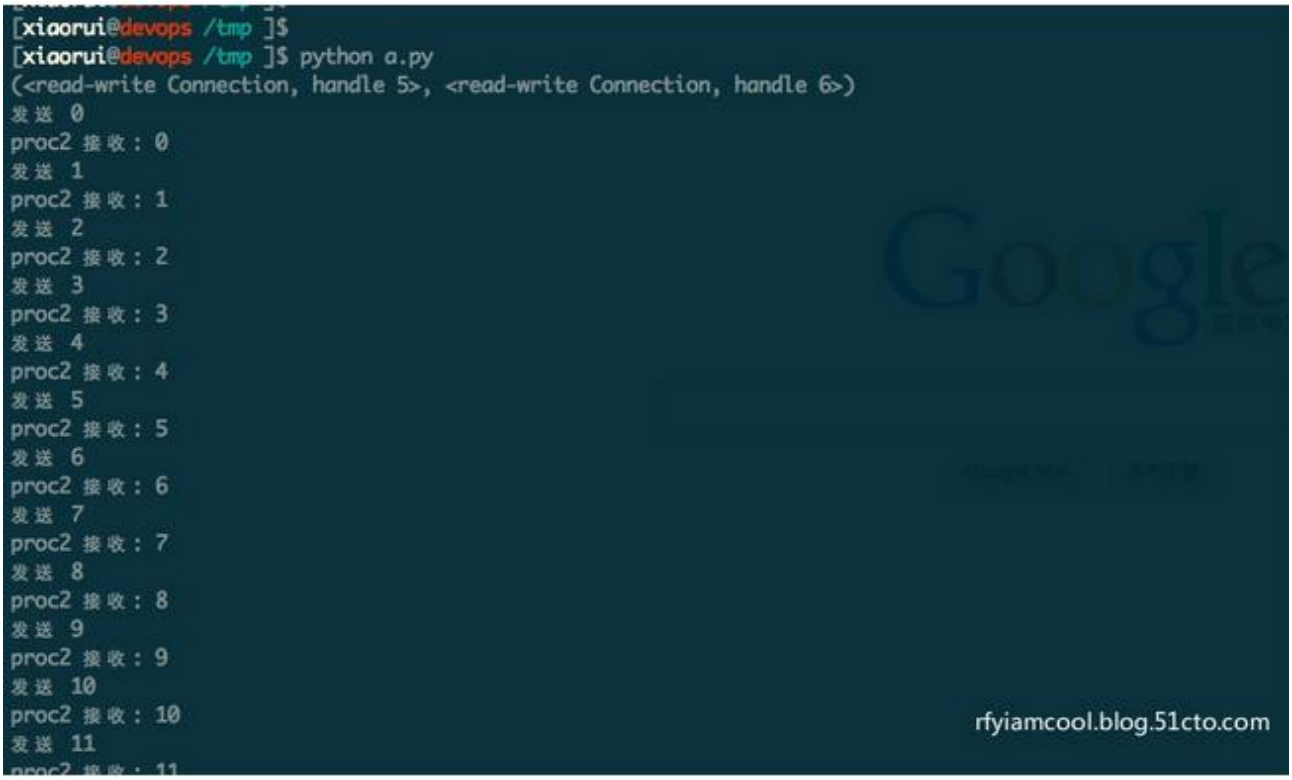
关于 python multiprocessing 进程通信的 pipe 和 queue 方式

作者：芮峰云 来源：http://rfyiamcool.blog.51cto.com/1030776/1549857

这两天温故了 python 的 multiprocessing 多进程模块，看到的 pipe 和 queue 这两种 ipc 方式，啥事 ipc？ ipc 就是进程间的通信模式，常用的一半是 socke，rpc，pipe 和消息队列等。

今个就再把 pipe 和 queue 搞搞。

```
#coding:utf-8
import multiprocessing
import time

def proc1(pipe):
while True:
    for i in xrange(10000):
        print "发送 %s"%i
        pipe.send(i)
        time.sleep(1)

def proc2(pipe):
while True:
    print 'proc2 接收:',pipe.recv()
    time.sleep(1)

def proc3(pipe):
while True:
    print 'proc3 接收:',pipe.recv()
    time.sleep(1)
# Build a pipe
pipe = multiprocessing.Pipe()
print pipe

# Pass an end of the pipe to process 1
p1   = multiprocessing.Process(target=proc1, args=(pipe[0],))
# Pass the other end of the pipe to process 2
p2   = multiprocessing.Process(target=proc2, args=(pipe[1],))


p1.start()
p2.start()
p1.join()
p2.join()
```



你来接收 或者是你来，我接收。 当然也可以做成双工的状态。

queue 的话，可以有更多的进程参与进来。用法和一些别的 queue 差不多。

看下官网的文档:

multiprocessing.Pipe([duplex])

Returns a pair (conn1, conn2) of Connection objects representing the ends of a pipe.

#两个 pipe 对象。用这两个对象，来互相的交流。

If duplex is True (the default) then the pipe is bidirectional. If duplex is False then the pipe is unidirectional: conn1 can only be used for receiving messages and conn2 can only be used for sending messages.

class multiprocessing.Queue([maxsize])

Returns a process shared queue implemented using a pipe and a few locks/semaphores. When a process first puts an item on the queue a feeder thread is started which transfers objects from a buffer into the pipe.

#队列的最大数

The usual Queue.Empty and Queue.Full exceptions from the standard library’ s Queue module are raised to signal timeouts.

Queue implements all the methods of Queue.Queue except for task_done() and join().

qsize()

Return the approximate size of the queue. Because of multithreading/multiprocessing semantics, this number is not reliable.

#队列的大小

Note that this may raise NotImplementedError on Unix platforms like Mac OS X where sem_getvalue() is not implemented.

empty()

Return True if the queue is empty, False otherwise. Because of multithreading/multiprocessing semantics, this is not reliable.

#是否孔了。 如果是空的，他回返回一个 True 的状态。

full()

Return True if the queue is full, False otherwise. Because of multithreading/multiprocessing semantics, this is not reliable.

#队列的状态是否满了。

put(obj[, block[, timeout]])

Put obj into the queue. If the optional argument block is True (the default) and timeout is None (the default), block if necessary until a free slot is available. If timeout is a positive number, it blocks at most timeout seconds and raises the Queue.Full exception if no free slot was available within that time. Otherwise (block is False), put an item on the queue if a free slot is immediately available, else raise the Queue.Full exception (timeout is ignored in that case).

#塞入队列，可以加超时的时间。

#文章原文: http://rfyiamcool.blog.51cto.com/1030776/1549857

put_nowait(obj)

Equivalent to put(obj, False).

#这里是不堵塞的

get([block[, timeout]])

Remove and return an item from the queue. If optional args block is True (the default) and timeout is None (the default), block if necessary until an item is available. If timeout is a positive number, it blocks at most timeout seconds and raises the Queue.Empty exception if no item was available within that time. Otherwise (block is False), return an item if one is immediately available, else raise the Queue.Empty exception (timeout is ignored in that case).

#获取状态

get_nowait()

Equivalent to get(False).

#不堵塞的 get 队列里面的数据

Queue has a few additional methods not found in Queue.Queue. These methods are usually unnecessary for most code:

close()

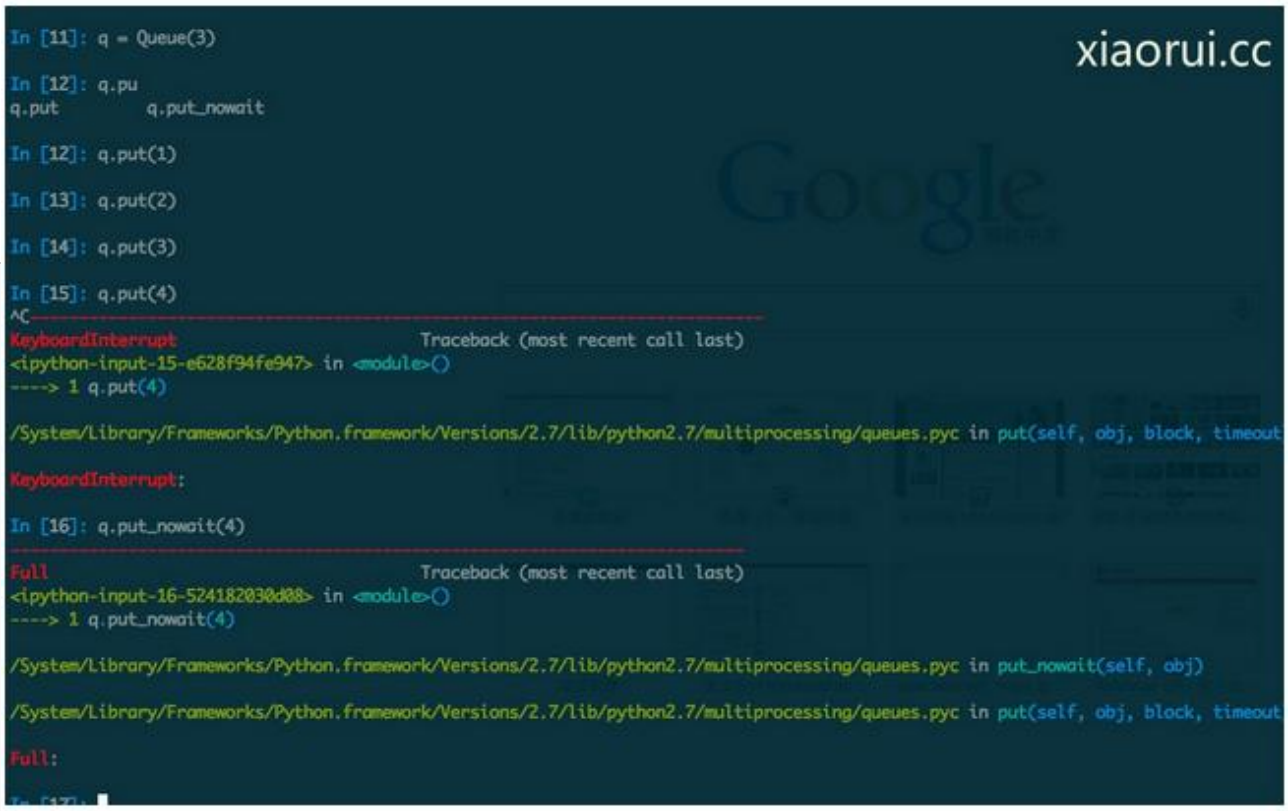
Indicate that no more data will be put on this queue by the current process. The background thread will quit once it has flushed all buffered data to the pipe. This is called automatically when the queue is garbage collected.

#关闭，省当前进程的资源。

我配置了 multiprocessing 队里长度是 3 个，然后当我放入的是第四个的时候， 会发现一只的堵塞，他是在等待，有人把数据 get 掉一个，那个时候 他才能继续的塞入 。如果用 put_nowait()的话，队列超出会立马会一个 error 的。

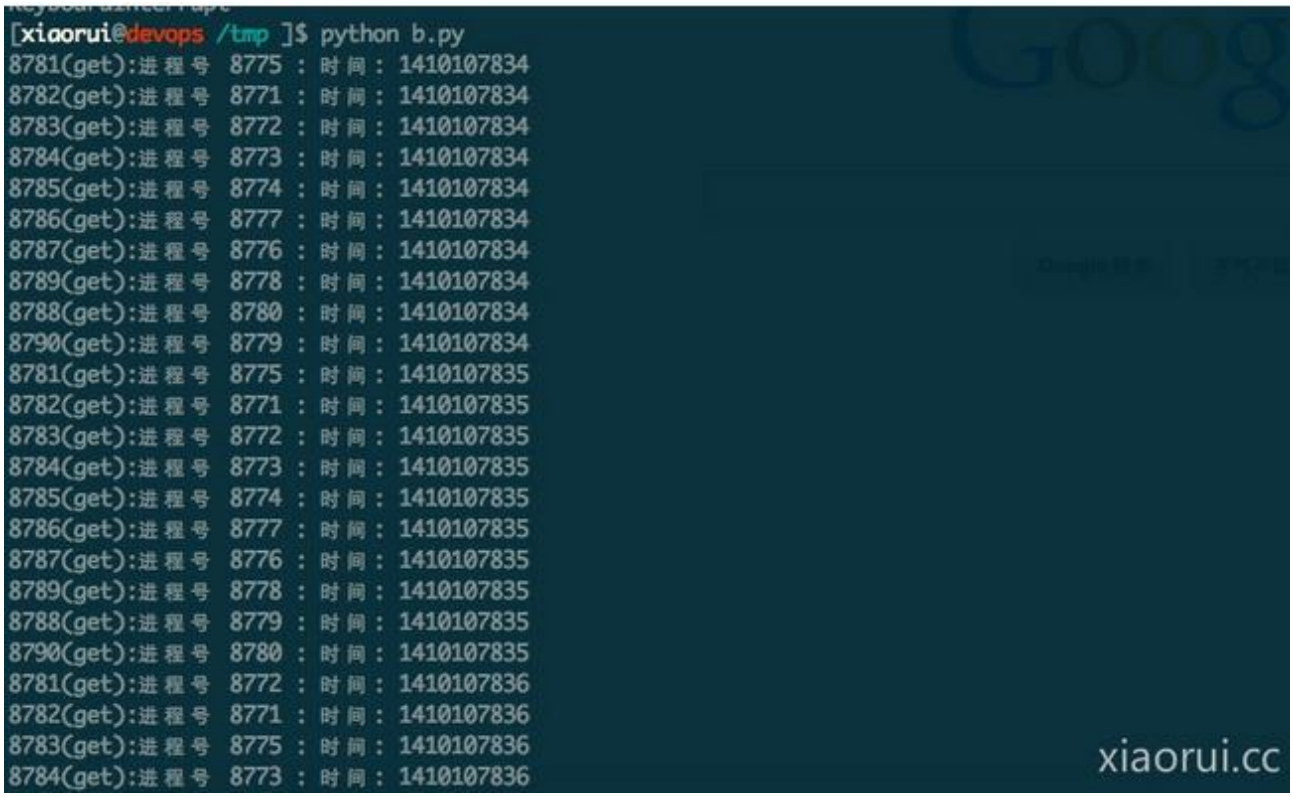
/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/multiprocessing/queues.pyc in put_nowait(self, obj)

/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/multiprocessing/queues.pyc in put(self, obj, block, timeout)




```
    info = queue.get()
#       lock.acquire()
    print (str(os.getpid()) + '(get):' + info)
#       lock.release()
    time.sleep(1)
#==================
# Main
record1 = []   # store input processes
record2 = []   # store output processes
lock  = multiprocessing.Lock()    # To prevent messy print
queue = multiprocessing.Queue(3)

# input processes
```

```
for i in range(10):
process = multiprocessing.Process(target=inputQ,args=(queue,))
process.start()
record1.append(process)

# output processes
for i in range(10):
process = multiprocessing.Process(target=outputQ,args=(queue,lock))
process.start()
record2.append(process)
```



一个 Web 页面的问题分析

> 作者：powertools 来源：http://powertoolsteam.blog.51cto.com/2369428/1547703

几个月之前我接到一个新的开发任务，要在一个旧的 Web 页面上面增添一些新的功能。在开发的过程中发现旧的代码中有很多常见的不合适的写法，结合这些问题，如何写出更好的，更规范的，更可维护的代码，就是这篇文章要阐述的内容。

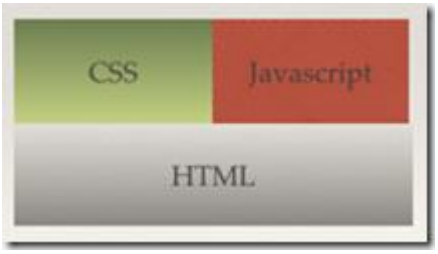
首先我查看了该 Web 页面的 HTML 代码，发现了一些典型的问题：

- HTML 页面中包含了很多 Javascript 和 CSS 代码
- HTML 页面中引用了大量的外部 Javascript 文件和 CSS 文件

接下来就这些问题，我们逐个讨论一下：

HTML 页面中包含了很多 Javascript 和 CSS 代码

一个正常的 Web 页面通常有以下三部分组成，HTML，CSS，Javascript，其中 HTML 是数据，CSS 负责样式，而 Javascript 负责交互，三者的关系如下图：



在构建 Web 页面的过程中，要尽量让这三者保持松耦合的关系，不要牵一发而动全身，一个层面小的改动需要改动另外两个层面。首先要从文件级别上隔离这三部分，在 HTML 中通过引入文件的方式导入 Javascript 和 CSS。

要做到三者的松耦合，开发中需要注意的地方又如下几点：

- 在 CSS 代码中不要包含 Javascript
- 在 Javascript 代码中不要包含 CSS
- 在 HTML 代码中不要包含 Javascript
- 在 Javascript 中不要包含 HTML

CSS 代码中不要包含 Javascript，指的是在 CSS 代码中慎用可计算的样式，如 IE 8 的 expression，CSS3 的 calc 等等，从使用角度来讲全是很强大，从代码维护的角度来看，不推荐使用。出现了 Bug 的时候，需要同时 Check Javascript 和 CSS 代码。

Javascript 代码中不要包含 CSS，我们经常需要在 Javascript 中去动态改变某一个 Dom 元素的样式，经常写出如下代码：

```
element.style.color = 'red';
```

这样的代码会导致当需求改变的时候，需要在 Javascript 代码中全文检索 red 关键字，深怕漏掉一点。推荐的做法如下：

```
//在 CSS 文件中定义样式类型.red-class{

  color: red;

}//Javascript 中改变样式 element.className += " red-class";// jQuery$(element).addClass("red-class");
```

在 Javascript 中操纵 Dom 对象的 Class 来改变样式，需求改变的时候，只需要调整 CSS 文件就可以了。

HTML 代码中不要包含 Javascript:

```
<input type="button" value="click me" id="mybutton" onclick="do()"/>
```

推荐使用下面的代码：

```
var btn = document.getElementById('mybutton');

btn.addEventListener("click", do);
```

Javascript 代码中不要包含 HTML：

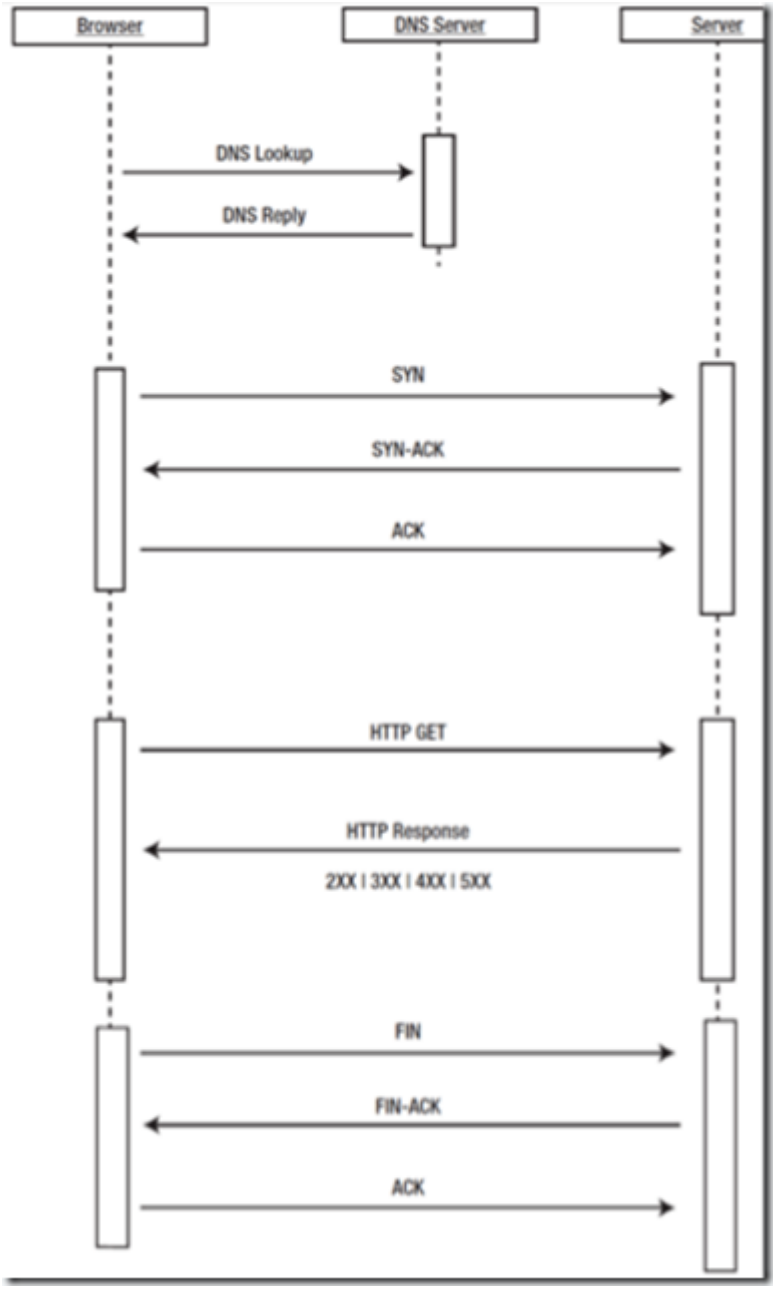
```
var div = document.getElementById("my-div");

div.innerHTML = "<h3>Error</h3><p>Invalid e-mail address.</p>";
```

在 Javascript 代码中完全隔绝 HTML 很难，这一点可以根据实际情况来权衡使用。Javascript 模版技术就是一种有效隔离 HTML 和 Javascript 代码的手段，如下是 jQuery Template 的用法：

```
// HTML<script id="bookTemplate" type="text/x-jQuery-tmpl">

    <div>

      <img src="BookPictures/${picture}" alt="" />

      <h2>${title}</h2>          price: ${formatPrice(price)}       </div>

 </script>// Javascript// Create an array of booksvar books = [{ title: "ASP.NET 4 Unleashed", price: 37.79, picture: "AspNet4Unleashed.jpg" }];// Render the books using the template$("#bookTemplate").tmpl(books).appendTo("#bookContainer");

function formatPrice(price) {     return "$" + price.toFixed(2);

}
```

HTML 页面中引用了大量的外部 Javascript 文件和 CSS 文件

HTML 页面中引用了大量的外部 Javascript 文件和 CSS 文件，我们知道每一个引用外部文件的<script>或者<style>都会引起一个 HTTP 请求，而一个 HTTP 请求的代价其实是很高昂的，下图是 HTTP 请求的整个过程：



首先要通过 DNS Server 把域名变为 IP，然后在浏览器与服务器之间建立 TCP 链接，建立 TCP 链接之后，浏览器向服务器发送 HTTP 请求，服务器处理完请求后，将结果返回给浏览器，最后关闭 TCP 链接。整个 HTTP 的请求的代价还是很大的，更多关于 HTTP 和 TCP 的信息，请参考：http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol，http://en.wikipedia.org/wiki/Transmission_Control_Protocol。

另外浏览器对于 HTTP 请求的并发数量是有限制的，每个浏览器不等，基本在 4 个左右。

当 HTML 页面中引用了大量的外部 Javascript 文件和 CSS 文件，我们可以考虑通过合并以及压缩 Javascript，CSS 文件来达到减少 HTTP 请求数量，以及 HTTP 结果的目的。

Grunt 是一个基于任务的 JavaScript 项目命令行构建工具，通过 Grunt 可以将多个文件合并成一个文件，并且进行压缩处理。Grunt 没有开发平台的限制，只要是前端项目，都可以使用 Grunt 来配置任务。Grunt 有着广泛的社区支持，有很多的现有的插件。

另外如果你是 ASP.NET 的项目的话，ASP.NET 4.5 加入了 Bundle，通过 Bundle 技术合并压缩 Javascript 和 CSS。关于 Bundle 技术可以参考 http://www.asp.net/mvc/tutorials/mvc-4/bundling-and-minification。

Javascript 代码全局变量

看完了 HTML 代码之后，又过了一下页面主要的 Javascript 代码，发现的问题主要是 Javascript 代码引入了太多的全局变量。JavaScript 全局变量在很小的程序中可能会带来方便，但随着程序越来越大，它很快变得难以处理。因为一个全局变量可以被程序的任何部分在任意时间改变，使得程序的行为被极大地复杂化。在程序中使用全局变量降低了程序的可靠性。

定义 Javascript 的方式有三种：

```
//  在所有函数外部使用 var 定义变量 var foo = 10;// 没有使用 var，直接声明变量 foo = 10;

window.foo = 10;
```

其中第二种隐式的声明了全局变量，尤其需要注意。

我们应该尽量少的引入全局变量，jQuery 也不过提供了两个全局变量：$, jQuery。那么有没有可能在注入 Javascript 到 HTML 页面之后，实现零个全局变量的引入？

通过立即执行函数可以达到这一点，参见下面的代码：

```
(function(win) {    "use strict";    var doc = win.document;    // declare other variables here

   // other code goes here}(window));
```

如果你需要将该对象返回，可以使用如下的方式：

```
var module1 = (function(){

        var _count = 0;

        var m1 = function(){

            //...             };

        var m2 = function(){

            //...             };

        return {

            m1 : m1,
```

```
            m2 : m2

        };

})();
```

这样的话只会引入一个全局变量 module1，而且该对象具有很好的封装性，其内部变量“_count”，在外部是无法访问的。 整个页面其实也还有一些其他小的问题，在这里就不一一赘述了。说了半天老代码的问题，其实没有对老代码有任何偏见，因为不论它是否美丑，都在为系统服务，都在产生价值。我们只是在追求更好的代码，更可读，更易维护的代码。

Oracle 11g rac 生产环境部署详录

作者：田逸 来源：http://sery.blog.51cto.com/10037/1546346

基本规 划

◎设备选型

1、服务器：Dell R620 两台。cpu 8 core，内存 64G，600G 15000 转 sas 硬盘，双电源，hba 卡一块，连接存储线缆一根（连接 hba 卡和共享存储）。

2、存储：dell MD3200 一台。双控制器，12 块 600G 15000 转 sas 硬盘。为追求最高可用性，使用的 raid 级别是 raid10.

3、交换机：华为 3com 两台，型号为 h3c S5048E。注意：网络端口最好是全千兆。

4、网线：2-3 米机制 6 类线数根。--曾遇到网线不够，问 IDC 机房要了根网管自己做的六类线，质量不行，结果导致 rac 节点之间心跳检查时好时坏。

5、辅助设备：vpn 及 kvm over ip 各一个

（1）小型简单的 vpn，DI-8200 上网行为管理认证路由器。价格 1000 多，便宜稳定。很适合拿来配 vpn，用于登录系统，管理各个服务器。

（2）kvm overip 型号为 ATEN cn8000。在系统不能远程登录的时候，打电话给 IDC 技术人员，让他们把这个设备连接到故障机，然后通过浏览器进行各种处理，如重装系统、查看屏幕信息、ctrl + Alt + Del 重启等。这跟直接去机房连显示器和键盘是一样的效果。我敢保证，打车去机房现场处理，一定没电话给 IDC 机房，让人给接上这个设备快.

◎网络规划

1、网卡绑定：2 个网卡绑定在一起，服务器的四个网卡，正好全部用上。即有效利用了带宽，又能增强可用性。

2、ip 地址分配：应用连接网络地址为 172.16.208.0/24，节点间心跳网络地址为 192.168.208.0/24。

3、网络连接及管理

（1）心跳网络与应用网络物理分离，各接一个交换机。

（2）服务器没有公网地址，必须通过 vpn 拨号，才能进行系统登录管理。

◎系统及所需工具

1、服务器操作系统：centos 5.9 64 位，定制安装系统，选件包括 xwindow 及 gnome。因安装 oracle 时需要图形用户界面，一个好的选择就是 vncserver。在进行正式部署前，最好确保安装好了 vncserver。一个定制好的系统的分区为：

```
[root@rac70 ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda6              29G   16G   12G  60% /
```

```
/dev/sda9          362G  215G  129G  63% /u01
/dev/sda8           15G  165M   14G   2% /tmp
/dev/sda7           19G   15G  3.9G  79% /home
/dev/sda5           29G  2.5G   25G  10% /usr
/dev/sda3           29G  321M   27G   2% /var
/dev/sda1           99M   19M   75M  21% /boot
tmpfs               32G   18G   15G  55% /dev/shm
```

2、oracle 软件：

（1）集群管理软件 linux.x64_11gR2_grid.zip，从 oracle 官网下载（需要注册为境外的电子邮件）

（2）数据库软件 linux.x64_11gR2_database，从 oracle 官网下载（需要注册为境外的电子邮件）

（3）存储管理软件 mdstoragemanager，从供货商获得光盘或者从 dell 的官网下载

（4）oracle asm 相关软件：oracleasm-2.6.18-308.el5-2.0.5-1.el5.x86_64.rpm、oracleasmlib-2.0.4-1.el5.x86_64.rpm、oracleasm-support-2.1.7-1.el5.x86_64.rpm。这三个软件，一定要跟操作系统内核版本相一致，否则不能进行后边的操作。

◎分区使用及文件系统

1、oracle 数据库及集群软件安装在本地硬盘/u01 分区。

（1）集群软件路径 /u01/app/grid

（2）数据库软件路径 /u01/app/oracle/product/11.2.0

（3）集群软件和数据库软件 base 路径为都为 /u01/app/oracle。因为软件安装路径（ORACLE_HOME）不能与 BASE 是同一个目录，因此这里弄得有点混乱，甚至有点别扭。幸运的是，这样令人疑惑的设置也能正常工作。希望将来的项目，可以提前把这样的设置规划得更规范些。打开一个安装好的环境，进入 ORACLE_BASE 目录，其子目录如下：

```
[root@db40 oracle]# ll
total 20
drwxr-x--- 3 oracle oinstall 4096 Mar  2 23:52 admin
drwxr-x--- 6 oracle oinstall 4096 Mar  2 23:52 cfgtoollogs
drwxr-xr-x 2 oracle oinstall 4096 Mar  2 23:20 checkpoints
drwxrwxr-x 5 oracle oinstall 4096 Mar  2 23:52 diag
drwxr-xr-x 3 oracle oinstall 4096 Mar  2 23:12 product
```

再进入目录 diag，有三个子目录存在：

```
[root@db40 diag]# ll
total 12
```

```
drwxr-x--- 3 oracle oinstall 4096 Mar  2 21:50 asm
drwxr-x--- 3 oracle oinstall 4096 Mar  2 23:52 rdbms
drwxr-xr-x 3 oracle oinstall 4096 Mar  2 21:53 tnslsnr
```

瞧，asm 文件系统、数据库以及监听器的告警日志都能在这里找到。如 asm 告警日志 /u01/app/oracle/diag/asm/+asm/+ASM2/alert/log.xml。

2、数据存储使用 oracle 自家 ASM（自动存储管理），划分三个 asm 磁盘组：OCR、FLASH、DATA（OCR 占据空间最小大约分配 500M；FLASH 次之，分配大概 300G；剩余的全部分配给 DATA）。其中：

◆OCR 存储集群注册信息

```
ASMCMD> pwd
+OCR/db1-scan
ASMCMD> ls
ASMPARAMETERFILE/
OCRFILE/
```

◆FLASH 存储归档日志及 rman 默认备份集

```
ASMCMD> cd FLASH
ASMCMD> ls
ZYZF/
ASMCMD> cd ZYZF
ASMCMD> ls
ARCHIVELOG/
BACKUPSET
```

◆DATA 存储 oracle 数据库元数据及用户数据

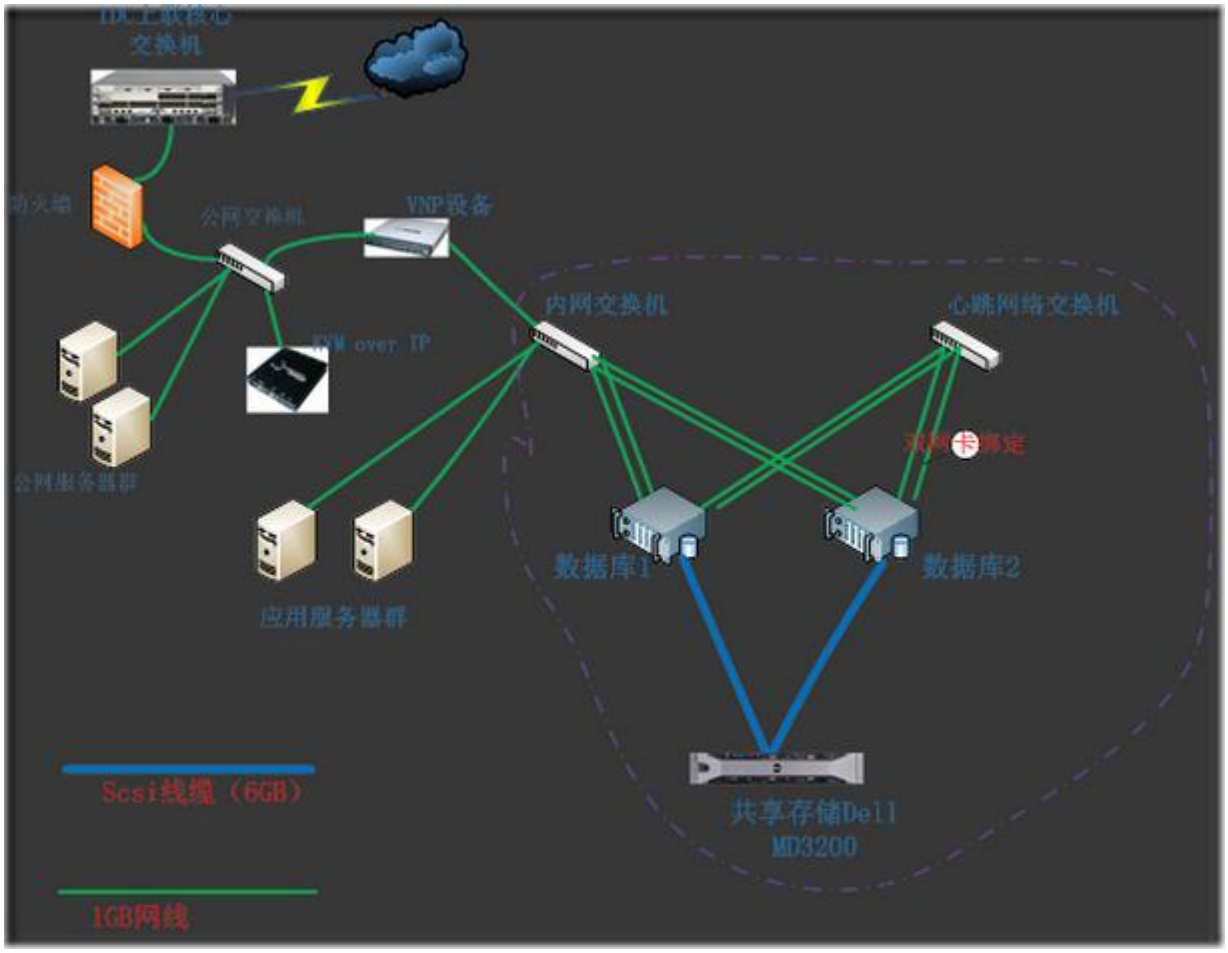
```
ASMCMD> cd DATA
ASMCMD> ls
ZYZF/
ASMCMD> cd ZYZF
ASMCMD> ls
CONTROLFILE/
DATAFILE/
ONLINELOG/
PARAMETERFILE/
TEMPFILE/
control01.ctl
control02.ctl
.....此处输出省略若干.....
```

redo07.log
redo08.log
spfilezyzf.ora

3、Dell MD3200 存储管理软件使用默认安装路径 /opt/dell/mdstoragesoftware。

4、asm 管理软件使用 rpm 包安装，安装文件自然满山放羊，到处都有：比如配置文件在 /etc/sysconfig/oracleasm，设备名在目录/dev/oracleasm/disks，可执行文件在这里 /etc/init.d/oracleasm。

◎拓扑图



执行步骤

1、 准备运行环境；
2、 准备及配置共享存储；
3、 准备数据库存储空间；
4、 安装集群工具 grid;
5、 安装和创建 oracle 数据库

特别说明：因为文档为后期撰写，为了描述更详尽，截图可能来自不同的运行环境，因此如果出现图片中 ip 地址与描述不一致的情况，请勿过多质疑。

（一）、准备运行环境

这里的运行环境环境主要包括远程桌面环境（我使用的是 vnc 服务）、安装软件所需的依赖工具、创建相关的帐号及目录、设置网络连接等。

◎设置网络（每个节点都需要设置）

◆修改文件 /etc/modprobe.conf,使其内容为：

```
alias scsi_hostadapter mpt2sas
alias eth0 tg3
alias eth1 tg3
alias eth2 tg3
alias eth3 tg3
alias scsi_hostadapter1 megaraid_sas
alias scsi_hostadapter2 ahci
alias scsi_hostadapter3 usb-storage
# Begin Dell MD Modification
options scsi_dh_rdac blacklist="DELL:MD3000,DELL:MD3000i"
# End Dell MD Modification
alias bond0 bonding
options bond0 miimon=100 mode=0
alias bond1 bonding
options bond1 miimon=100 mode=0
```

◆修改网络接口文件，使其内容如下（为了描述方便，数个文件一起贴出）

```
[root@rac70 network-scripts]# more ifcfg-eth0
# Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
DEVICE=eth0
#HWADDR=E0:DB:55:20:E7:10
ONBOOT=yes
TYPE=Ethernet
USERCTL=no
MASTER=bond0
SLAVE=yes
TYPE=Ethernet

[root@rac70 network-scripts]# more ifcfg-eth1
# Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
DEVICE=eth1
BOOTPROTO=none
```

```
#HWADDR=e0:db:55:20:e7:11
ONBOOT=yes
TYPE=Ethernet
USERCTL=no
MASTER=bond0
SLAVE=yes
TYPE=Ethernet

[root@rac70 network-scripts]# more ifcfg-eth2
# Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
DEVICE=eth2
BOOTPROTO=none
#HWADDR=e0:db:55:20:e7:12
ONBOOT=yes
TYPE=Ethernet
USERCTL=no
MASTER=bond1
SLAVE=yes
TYPE=Ethernet

[root@rac70 network-scripts]# more ifcfg-eth3
# Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
DEVICE=eth3
#HWADDR=E0:DB:55:20:E7:13
ONBOOT=yes
HOTPLUG=no
BOOTPROTO=none
TYPE=Ethernet
MASTER=bond1
SLAVE=yes
TYPE=Ethernet
```

◆创建新的网络配置文件，其完整内容如下：

```
[root@rac70 network-scripts]# more ifcfg-bond0
# Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
DEVICE=bond0
BOOTPROTO=none
```

```
BROADCAST=172.16.208.255
IPADDR=172.16.208.70
NETMASK=255.255.255.0
NETWORK=172.16.208.0
ONBOOT=yes
GATEWAY=172.16.208.201
TYPE=Ethernet

[root@rac70 network-scripts]# more ifcfg-bond1
# Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
DEVICE=bond1
BOOTPROTO=none
ONBOOT=yes
DHCP_HOSTNAME=oracle70
TYPE=Ethernet
USERCTL=no
IPADDR=192.168.208.70
NETMASK=255.255.255.0
```

◆追加下面的行到/etc/rc.local 文件

```
modprobe bonding miimon=100 mode=0
```

配置完毕后，重启一下系统，检查配置的正确性及联通性。正确配置的输出如下图所示：

```
[root@rac70 network-scripts]# ifconfig -a
bond0     Link encap:Ethernet  HWaddr E0:DB:55:20:E7:10
          inet addr:172.16.208.70  Bcast:172.16.208.255  Mask:255.255.255.0
          inet6 addr: fe80::e2db:55ff:fe20:e710/64 Scope:Link
          UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
          RX packets:1816790422 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1789899980 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:549300301861 (511.5 GiB)  TX bytes:1389874576410 (1.2 TiB)

bond0:1   Link encap:Ethernet  HWaddr E0:DB:55:20:E7:10
          inet addr:172.16.208.74  Bcast:172.16.208.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1

bond0:2   Link encap:Ethernet  HWaddr E0:DB:55:20:E7:10
          inet addr:172.16.208.72  Bcast:172.16.208.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1

bond1     Link encap:Ethernet  HWaddr E0:DB:55:20:E7:12
          inet addr:192.168.208.70  Bcast:192.168.208.255  Mask:255.255.255.0
          inet6 addr: fe80::e2db:55ff:fe20:e712/64 Scope:Link
          UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
          RX packets:2703701331 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2594888233 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1961245879185 (1.7 TiB)  TX bytes:1942688172201 (1.7 TiB)

eth0      Link encap:Ethernet  HWaddr E0:DB:55:20:E7:10
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:789497216 errors:0 dropped:0 overruns:0 frame:0
          TX packets:894951420 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:245984374158 (229.0 GiB)  TX bytes:694916167061 (647.1 GiB)
          Interrupt:210 Memory:d91a0000-d91b0000

eth1      Link encap:Ethernet  HWaddr E0:DB:55:20:E7:10
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
```

◎配置 vnc 服务(每个 rac 节点都需要配置)

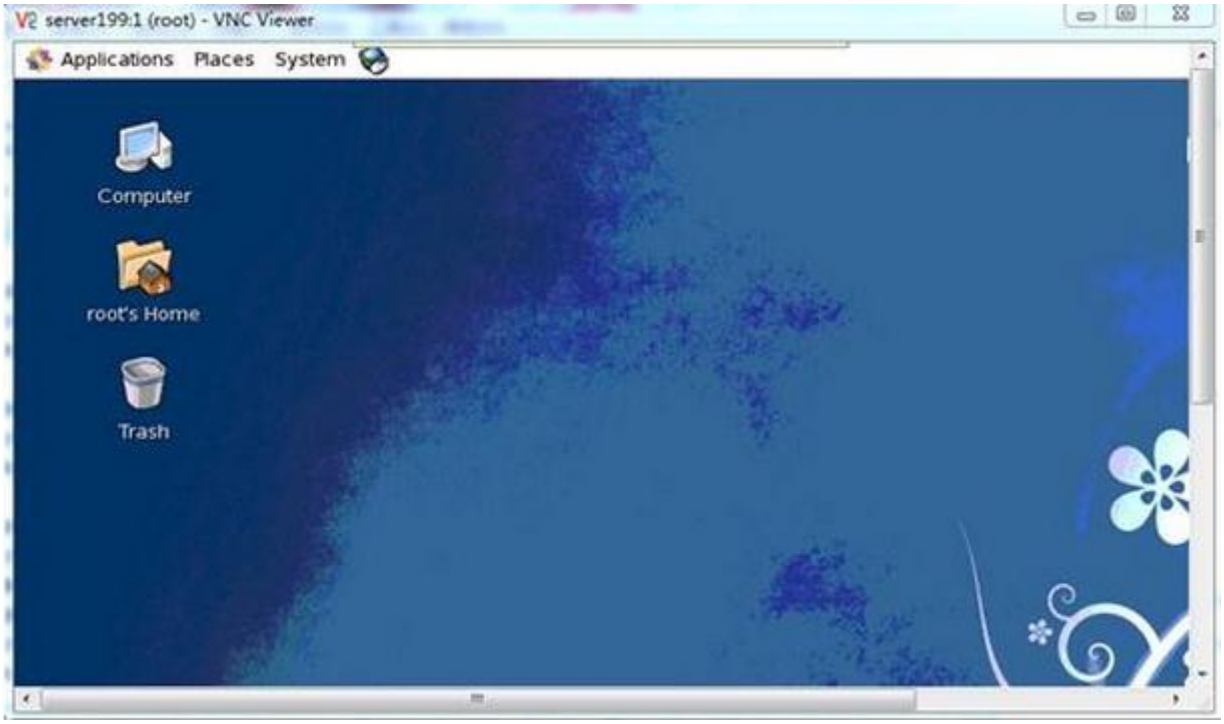
1、确保安装了 vncserver 软件。测试方法：任意路径执行命令 vncserver，如果命令不存在，则运行命令 yum install vncserver 安装一下。

2、执行 vncserver 启动服务，如果是第一次启动，需要设置 vnc 服务的密码，这里建议设置跟系统 root 相同的密码（要用复杂密码哟！）。

3、修改文件/root/.vnc/xstart,使其内容为：

```
#!/bin/sh

# Uncomment the following two lines for normal desktop:
unset SESSION_MANAGER
exec /etc/X11/xinit/xinitrc

[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
vncconfig -iconic &
xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
#twm &
gnome start&
```

4、重启系统，再启动 vncserver，然后在 windows 机器启用 vnc 客户端，远程连接，看是否能出现远程服务器的图形桌面。



◎创建安装软件所需的帐号、目录、依赖软件等。为了节省时间和避免出错，我把这些步骤写成一个粗糙的脚本，执行一次就可以了。脚本的内容如下：

```
[root@db40 oracle_rac]# more oracle_rac_rep.sh
#!/bin/bash
#writed by sery 2012-05-16

########################################
#install depending packages            #
########################################
yum -y install binutils compat-libstdc++-33 elfutils-libelf elfutils-libelf-devel
glibc \ glibc-common glibc-devel gcc gcc-c++ libaio-devel \
libaio libgcc libstdc++ libstdc++-devel make sysstat unixODBC unixODBC-
devel \
pdksh numactl-devel glibc-headers

########################################
#add groups,user and create dir        #
########################################
/usr/sbin/groupadd -g 501 oinstall
/usr/sbin/groupadd -g 502 dba
/usr/sbin/groupadd -g 504 asmadmin
/usr/sbin/groupadd -g 506 asmdba
/usr/sbin/groupadd -g 507 asmoper

useradd -u 1000 -g oinstall -G dba,asmdba oracle
useradd -u 1006 -g oinstall -G asmadmin,asmdba,asmoper grid

mkdir /u01/app/
chown -R grid:oinstall /u01/app/
chmod -R 775 /u01/app/

mkdir -p /u01/app/oraInventory
chown -R grid:oinstall /u01/app/oraInventory
chmod -R 775 /u01/app/oraInventory

mkdir -p /u01/app/grid
```

```
mkdir -p /u01/app/oracle

chown -R grid:oinstall /u01/app/grid
chown -R oracle:oinstall /u01/app/oracle

chmod -R 775 /u01/app/oracle
chmod -R 775 /u01/app/grid

################################################
#modify  sysctl.conf                           #
################################################
cat >> /etc/sysctl.conf <<done
fs.file-max = 6815744
kernel.shmall = 2097152
#kernel.shmmax = 536870912
kernel.shmmni = 4096
kernel.sem = 250 32000 100 128
net.ipv4.ip_local_port_range = 9000 65500
net.core.rmem_default = 262144
net.core.rmem_max = 4194304
net.core.wmem_default = 262144
net.core.wmem_max = 1048576
fs.aio-max-nr = 1048576
done

sysctl -p

################################################
#modify  /etc/security/limits.conf            #
################################################
cat >> /etc/security/limits.conf << done
grid soft nproc 2047
grid hard nproc 16384
grid soft nofile 1024
grid hard nofile 65536
oracle soft nproc 2047
```

```
oracle hard nproc 16384
oracle soft nofile 1024
oracle hard nofile 65536
done

################################################
#modify /etc/pam.d/login                       #
################################################
echo "session    required     pam_limits.so">>/etc/pam.d/login

################################################
# setting user oracle env                      #
################################################
cat >> /home/oracle/.bash_profile <<done
export ORACLE_SID=zyzf1
export ORACLE_UNQNAME=zyzf1
export ORACLE_base=/u01/app/oracle
export ORACLE_HOME=/u01/app/oracle/product/11.2.0
export PATH=$ORACLE_HOME/bin:$PATH
done

################################################
#setting user grid env                         #
################################################
cat >> /home/grid/.bash_profile <<done
export ORACLE_SID=+ASM1
export ORACLE_BASE=/u01/app/oracle
export ORACLE_HOME=/u01/app/grid
export PATH=$ORACLE_HOME/bin:$PATH
done
```

◎设置相关主机名及 ip 映射（每个节点都要进行）

修改文件/etc/hosts，使其内容为：

```
[root@db40 oracle_rac]# more /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       db40    localhost.localdomain localhost
```

```
::1             localhost6.localdomain6 localhost6

# Pubic
172.16.5.40       db40
172.16.5.50       db50
# Private
10.16.5.40       db40-priv
10.16.5.50       db50-priv
# Virtual
172.16.5.41       db40-vip
172.16.5.51       db50-vip
# SCAN
172.16.5.55       db1-scan

#dghost
172.16.5.60       datadg
```

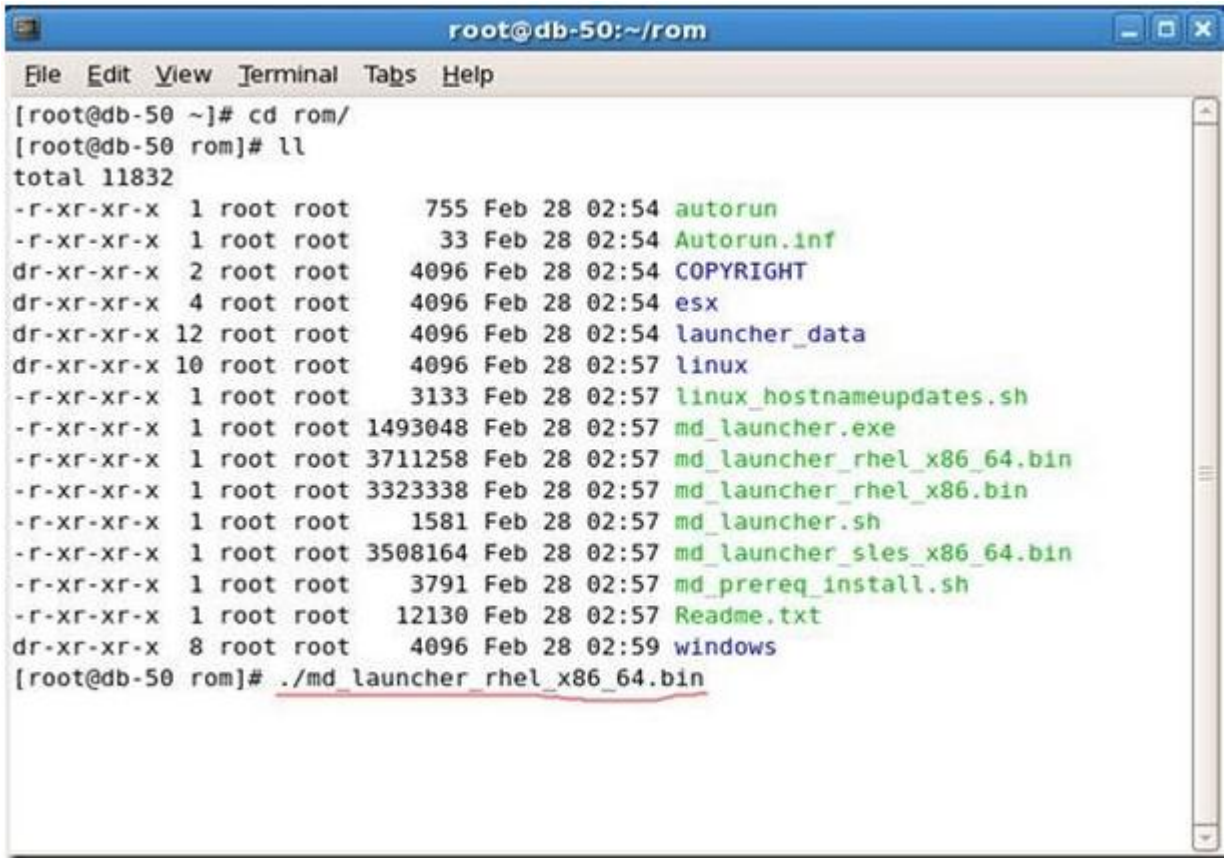
（二）、准备及配置共享存储

◆确保服务器与存储设备做好了物理连接：通过查看设备指示灯获得信息。正常情况下，hba 卡的指示灯、存储设备连线处的指示灯，都应该显示绿色。

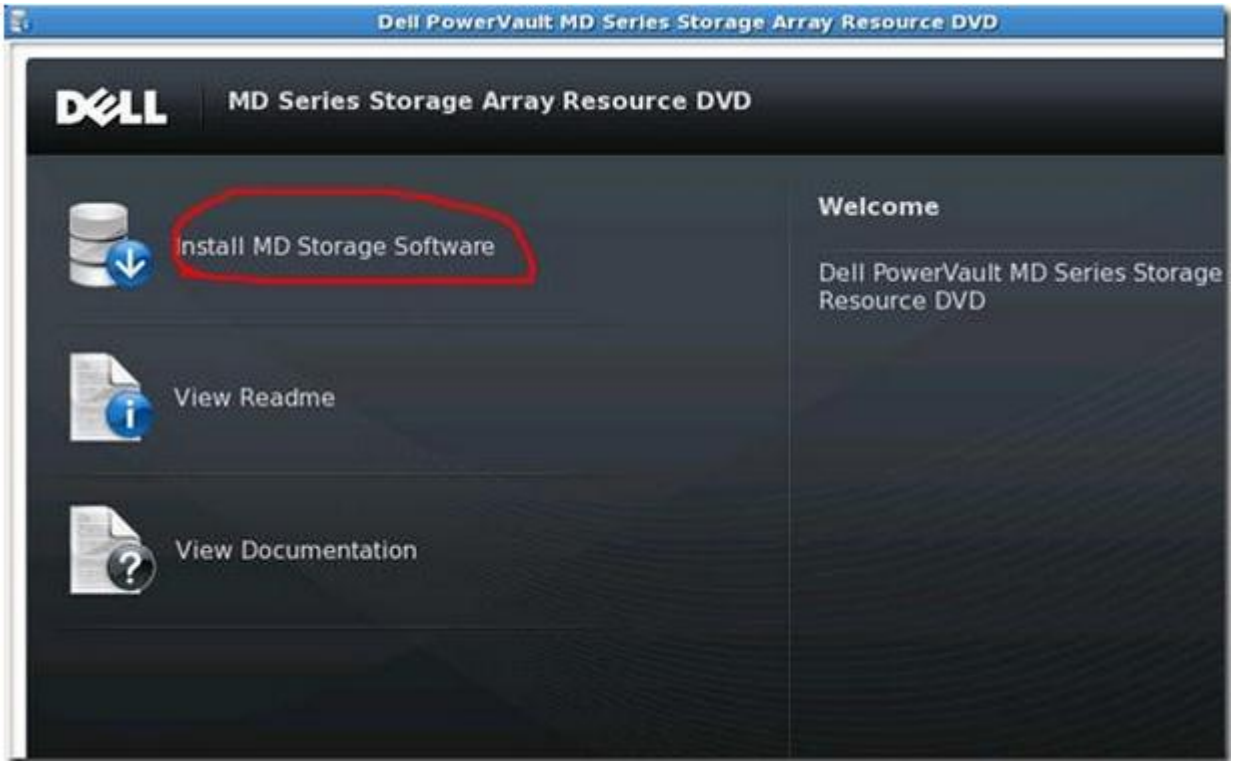
◆主机安装存储管理软件 powervault。注意：每个 rac 节点都需要安装，否则未安装的节点将不能发现共享磁盘。具体的过程如下：

◎远程 vnc 连接服务器

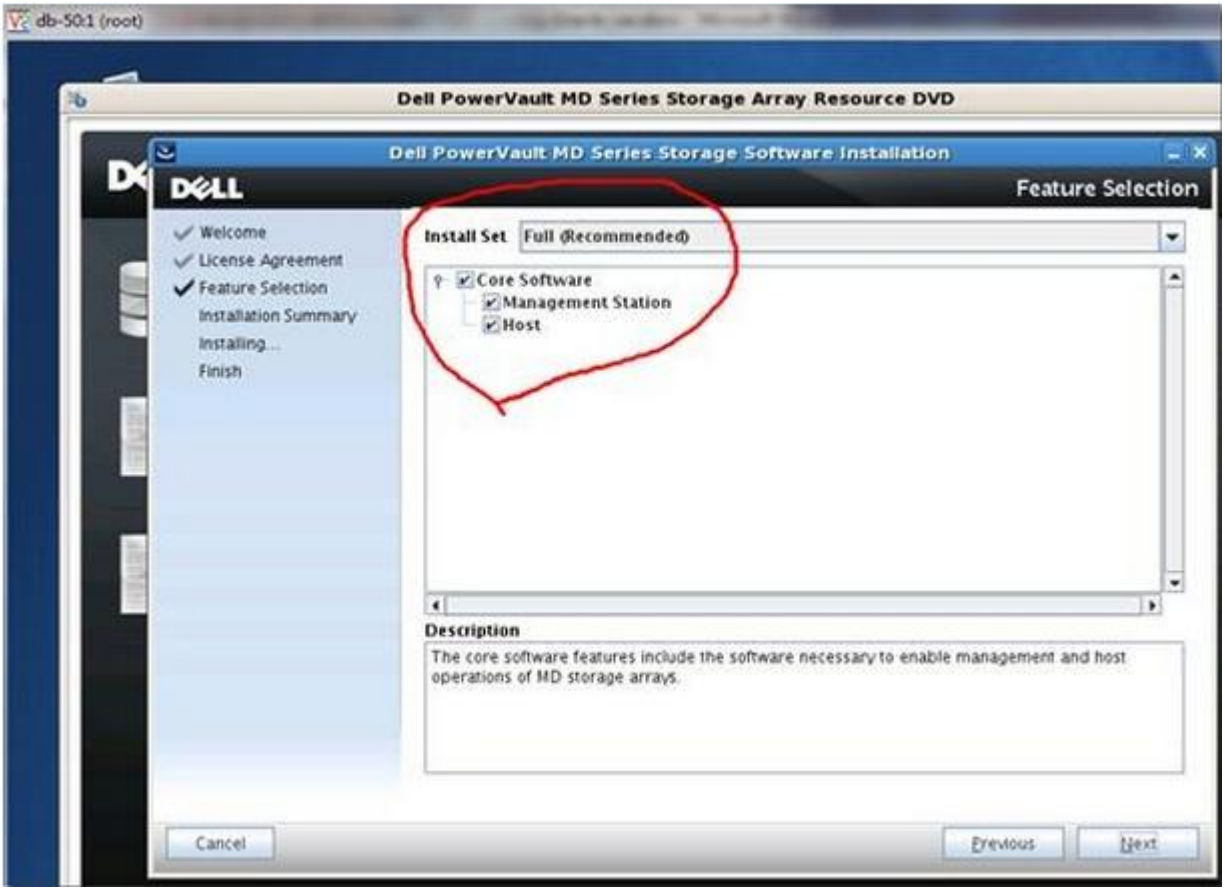
◎进入软件目录，执行命令./md_launcher_rhel_x86_64.bin



弹出安装窗口；



鼠标点击 Install MD Storage Software；

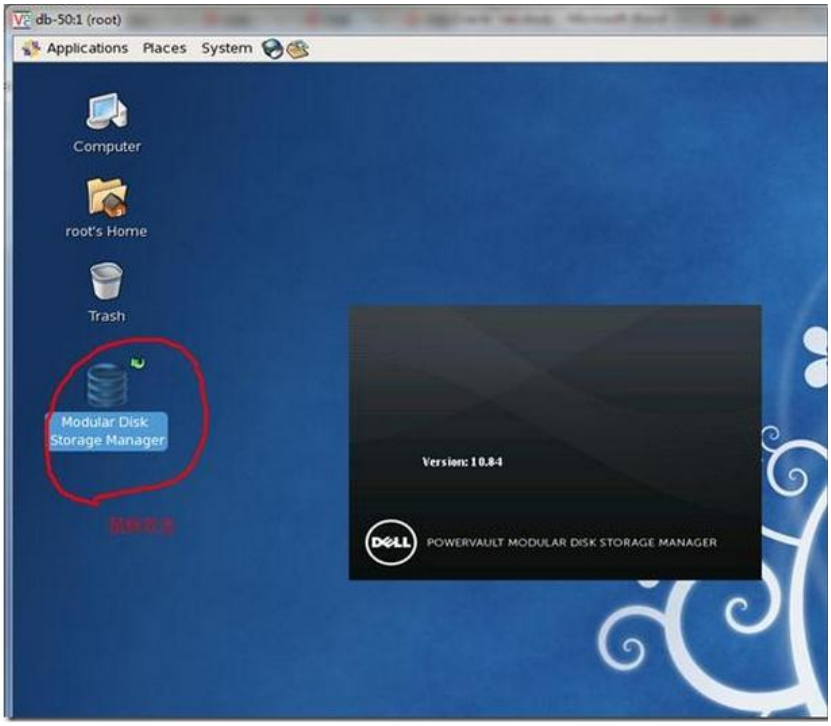

db-50:1 (root)
Dell PowerVault MD Series Storage Array Resource DVD
Dell PowerVault MD Series Storage Software Installation
DELL
Feature Selection
Welcome
License Agreement
Feature Selection
Installation Summary
Installing...
Finish
Install Set
Full (Recommended)
Core Software
Management Station
Host
Description
The core software features include the software necessary to enable management and host operations of MD storage arrays.
Cancel
Previous
Next

选默认的情况就行；


db-50:1 (root)
Dell PowerVault MD Series Storage Array Resource DVD
Dell PowerVault MD Series Storage Software Installation
DELL
Choose MD Series Model
Welcome
License Agreement
Feature Selection
Installation Summary
Installing...
Finish
Select the MD Storage Array connection type that will be managed by MD Storage Manager on this host or the arrays that will be accessed by this host.
Fibre Channel (MD3600f, MD3620f, MD3660f)
iSCSI (MD3200i, MD3220i, MD3260i, MD3600i, MD3620i, MD3660i)
SAS (MD3200, MD3220, MD3260)
Description
Configures data communication between the host server and iSCSI and/or SAS storage arrays you select. Multiple MD models can be selected (excluding Fibre Channel).
NOTE: Regardless of the model you select here, all storage arrays can be managed from this host server using MD Storage Manager.
Clear All
Cancel
Previous
Next

选择 sas（我用的是 MD3200，正好匹配），后边的安装就很容易了，这里不在赘述。安装完系统后，将

在桌面生成图标，点击图表，就可以进行存储管理。

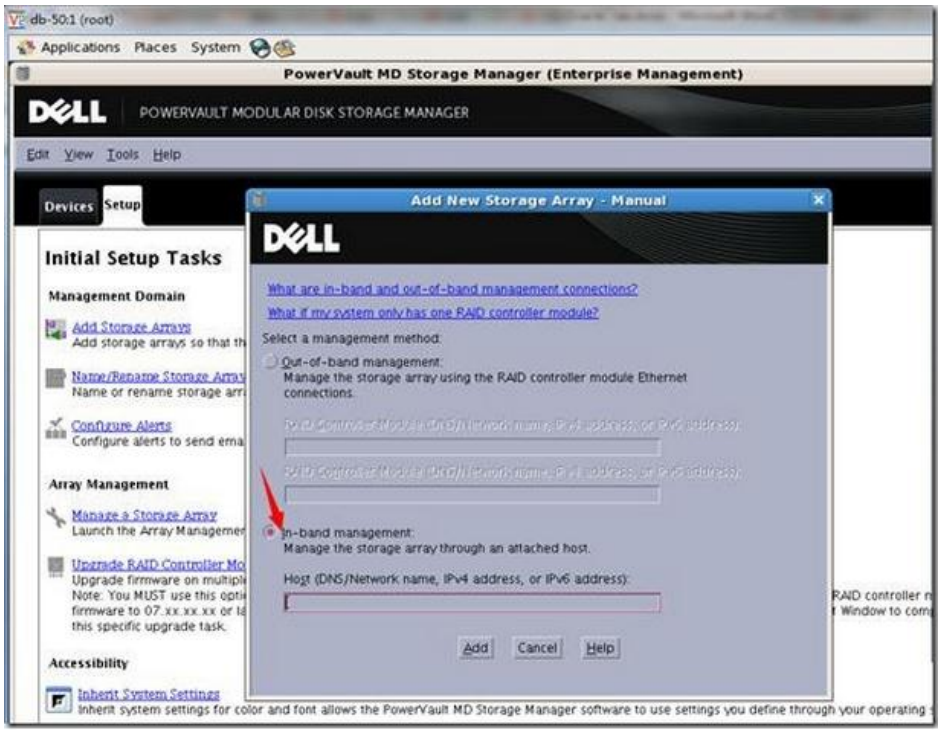


如果不幸出现如下错误，请修改/etc/hosts 文件解决

◎配置共享存储(在一个节点上执行即可)

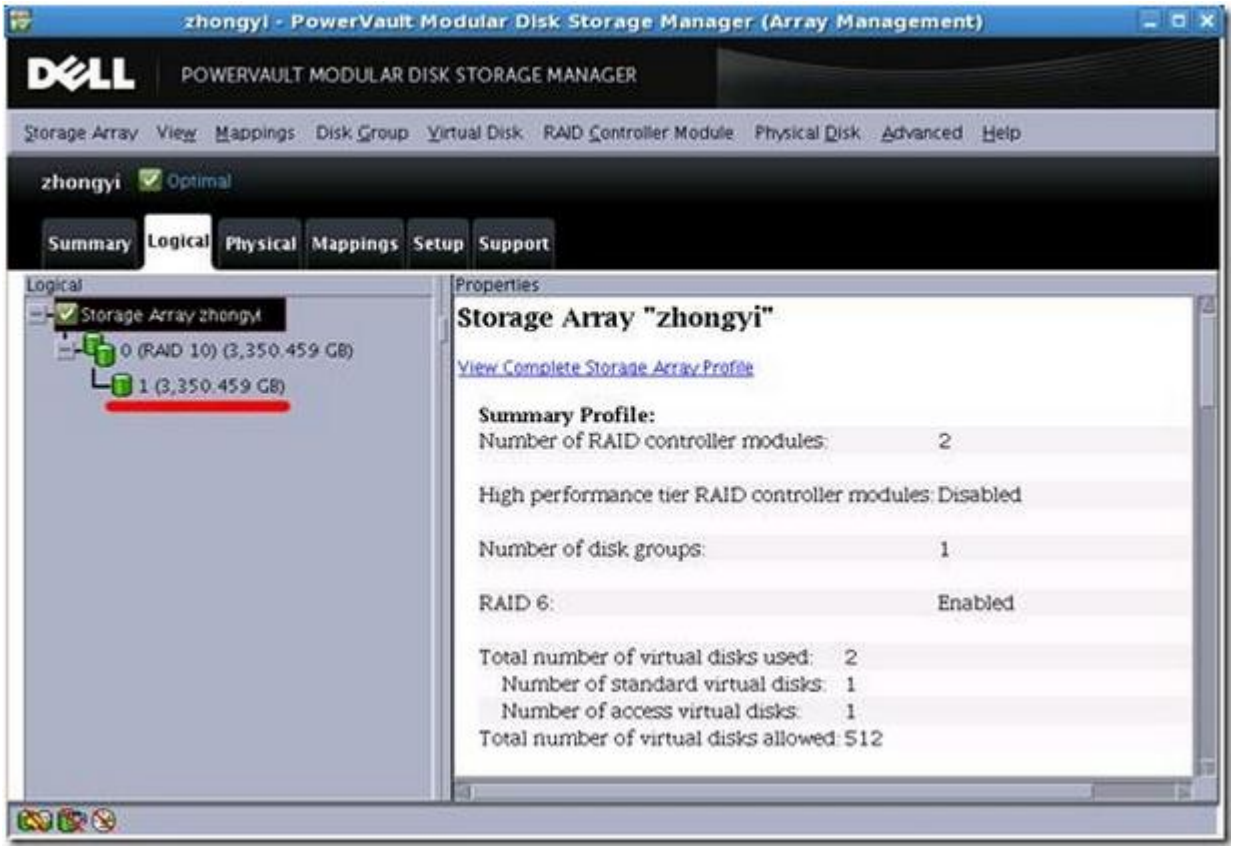
1、双击图标，启动存储管理工具

2、选择手动 Manual，然后选择带内管理

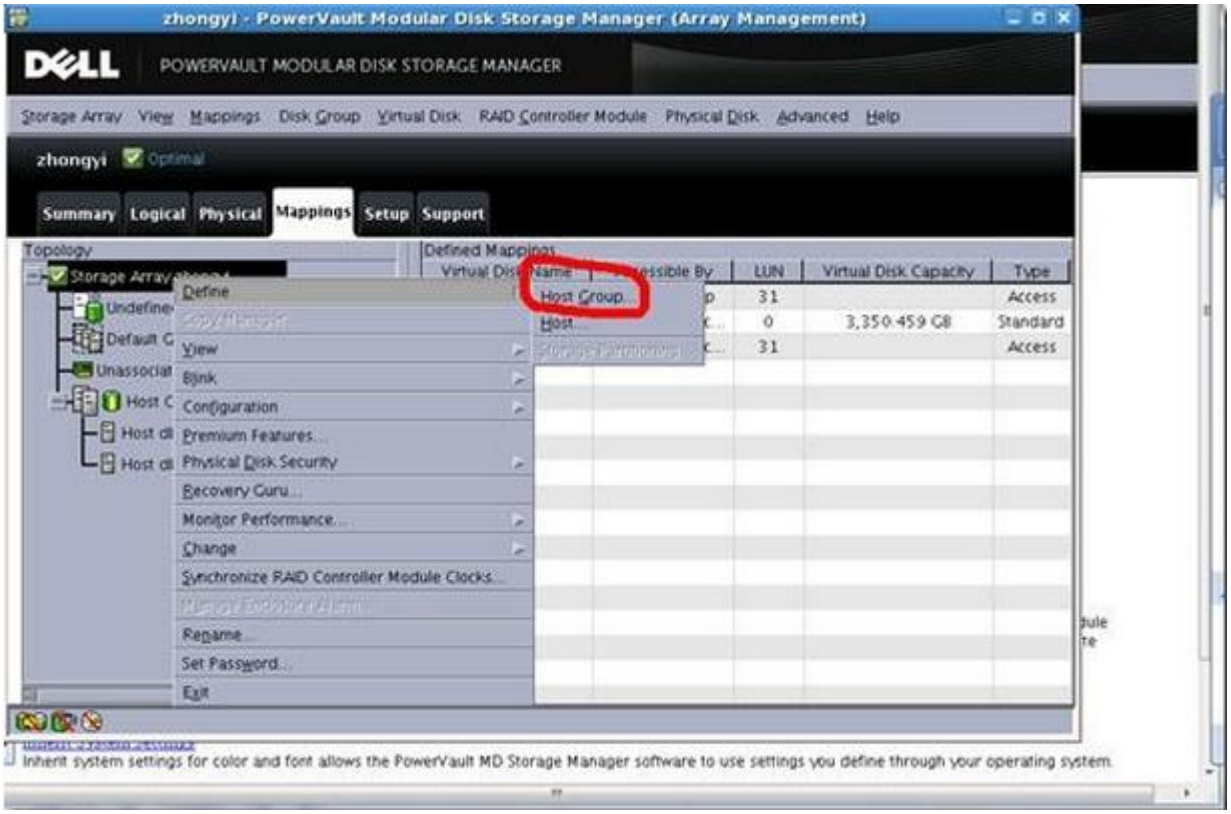


3、此处输入连接磁盘阵列的主机 ip 地址

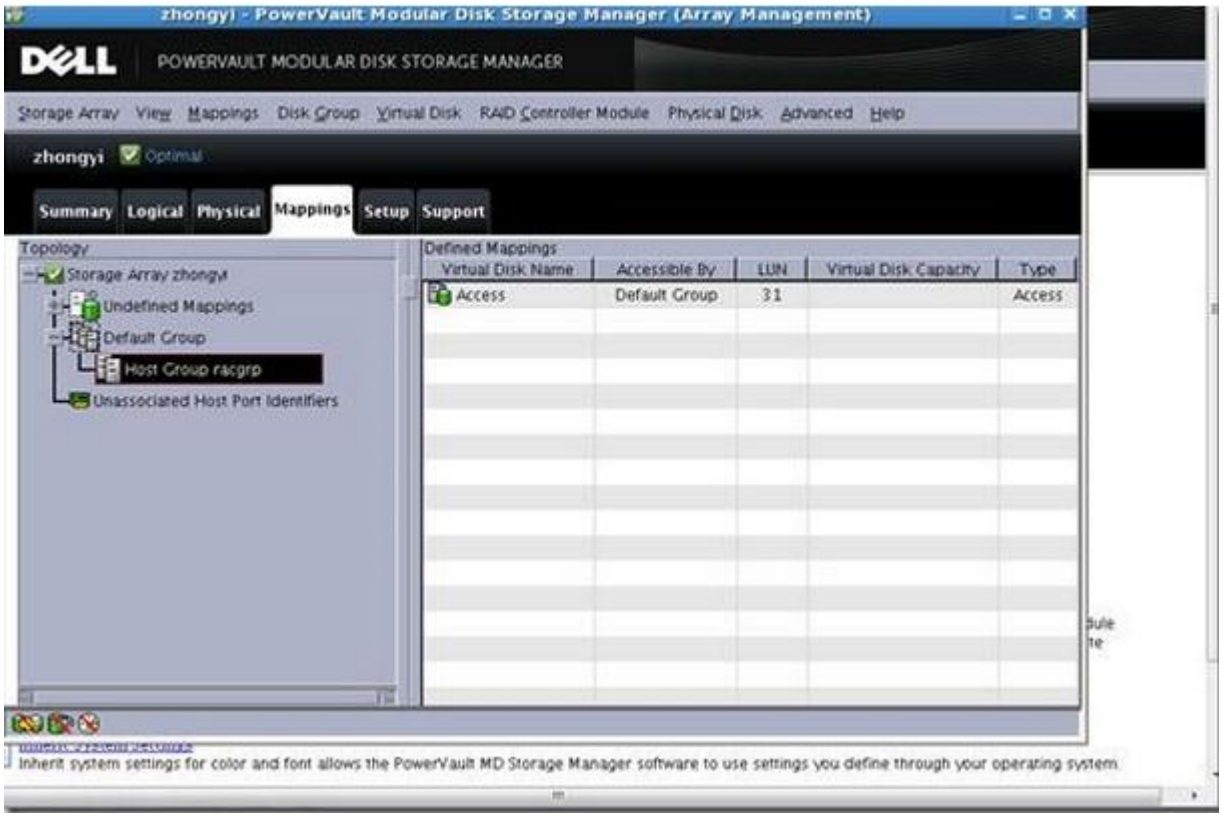
4、创建磁盘组，选择 raid 级别为 RAID10，把 12 个盘都选择上，这个过程很费时间



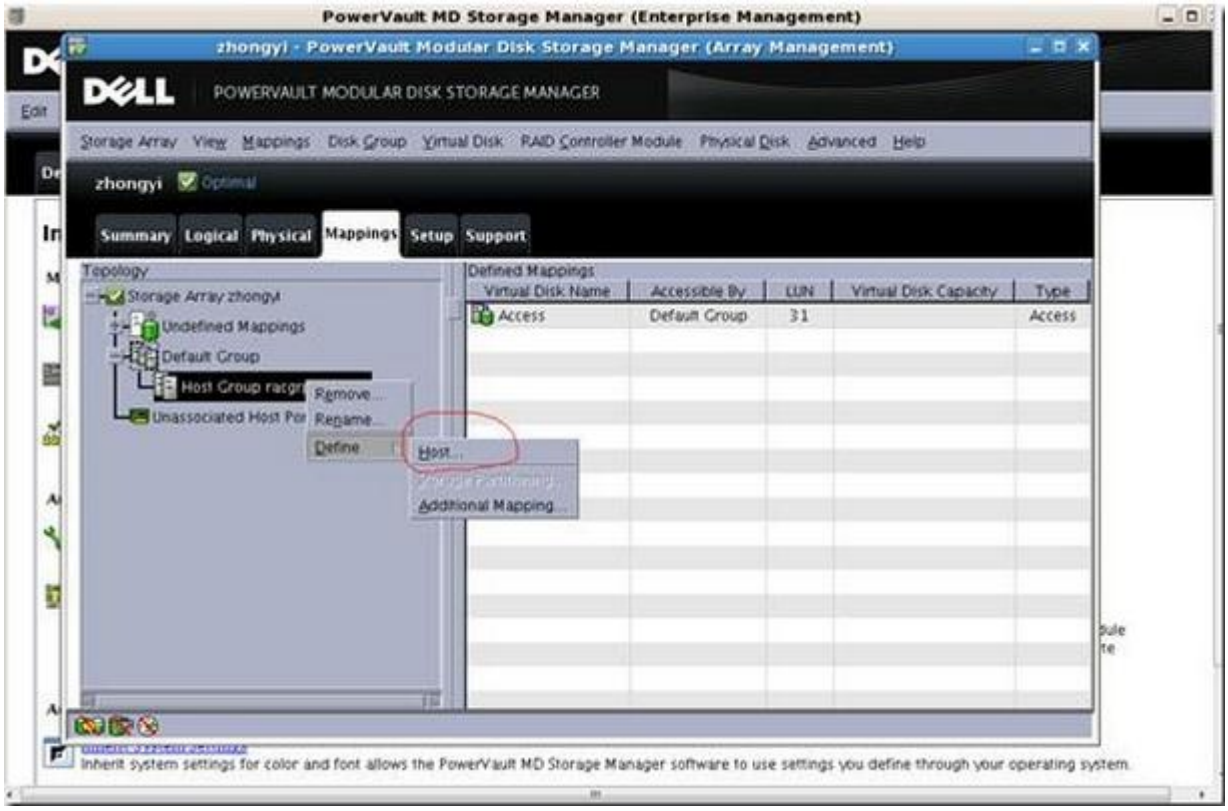
5、虚拟磁盘映射到主机，目的是要连接到阵列的主机都能访问到虚拟磁盘组



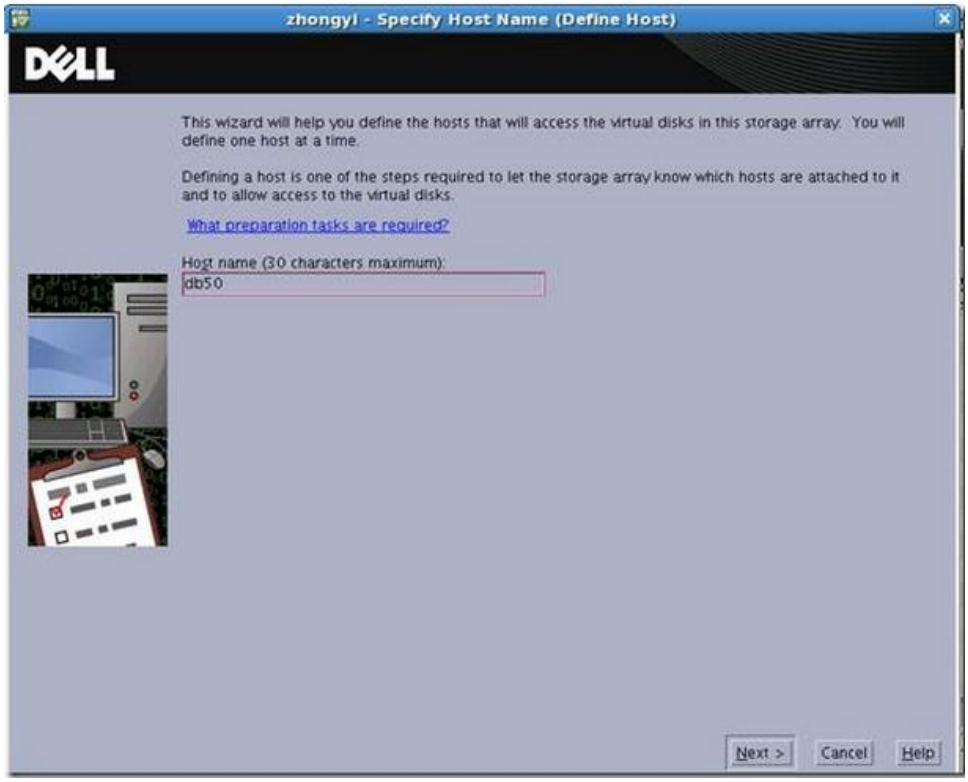
6、填入一个容易标识的名称



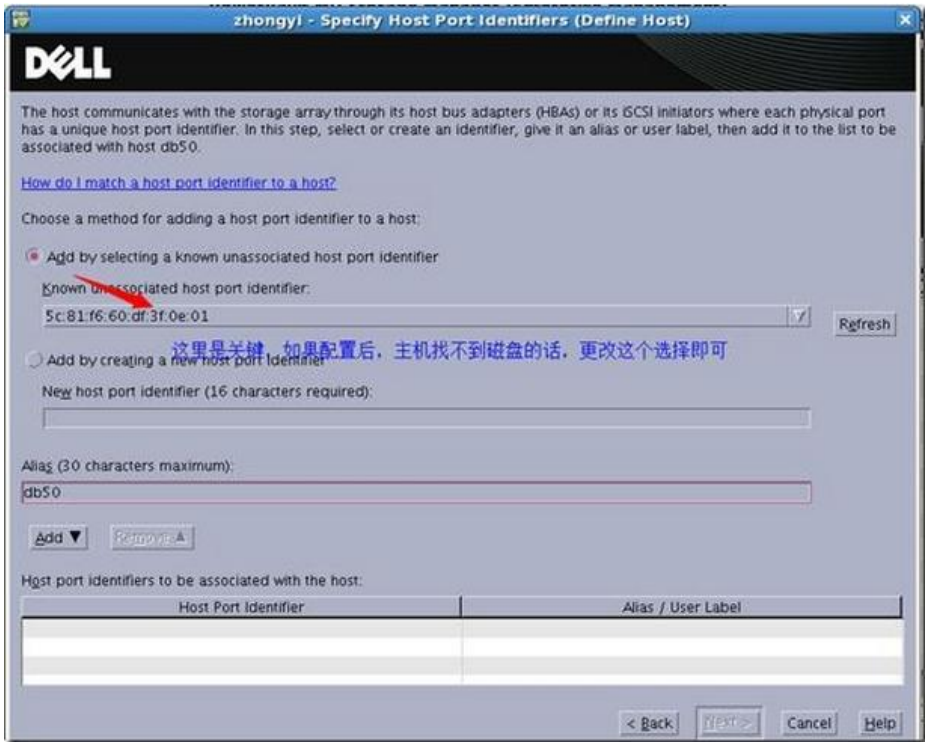
7、鼠标右键点击刚创建的主机组名称，调出主机定义



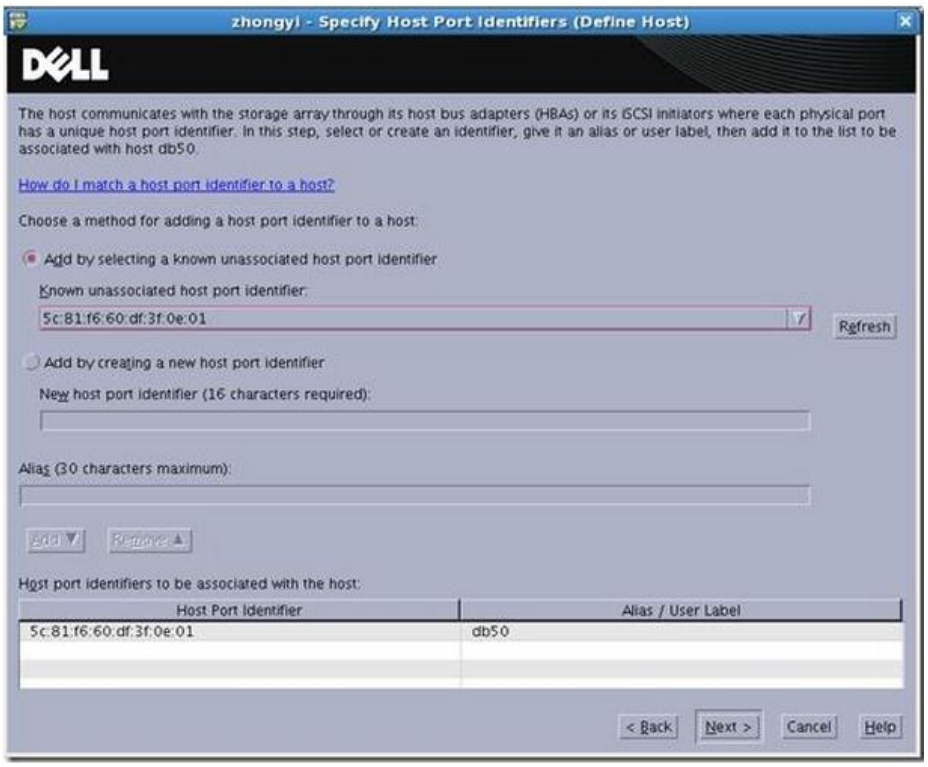
8、输入主机名（以/etc/hosts 为准）



9、选择连接端口，这里需要注意，弄错了，主机识别不到磁盘组的

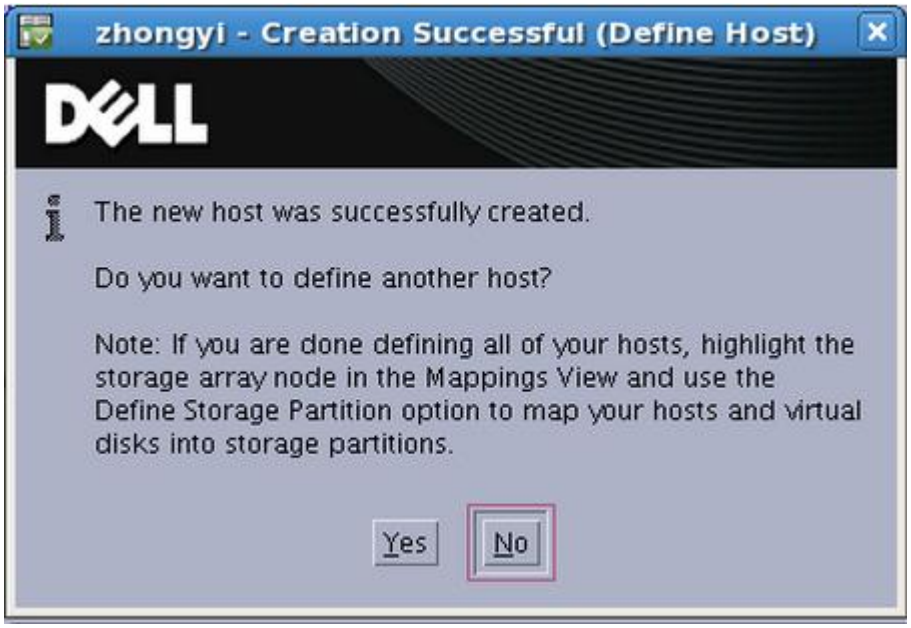


10、鼠标点击按钮”Add”

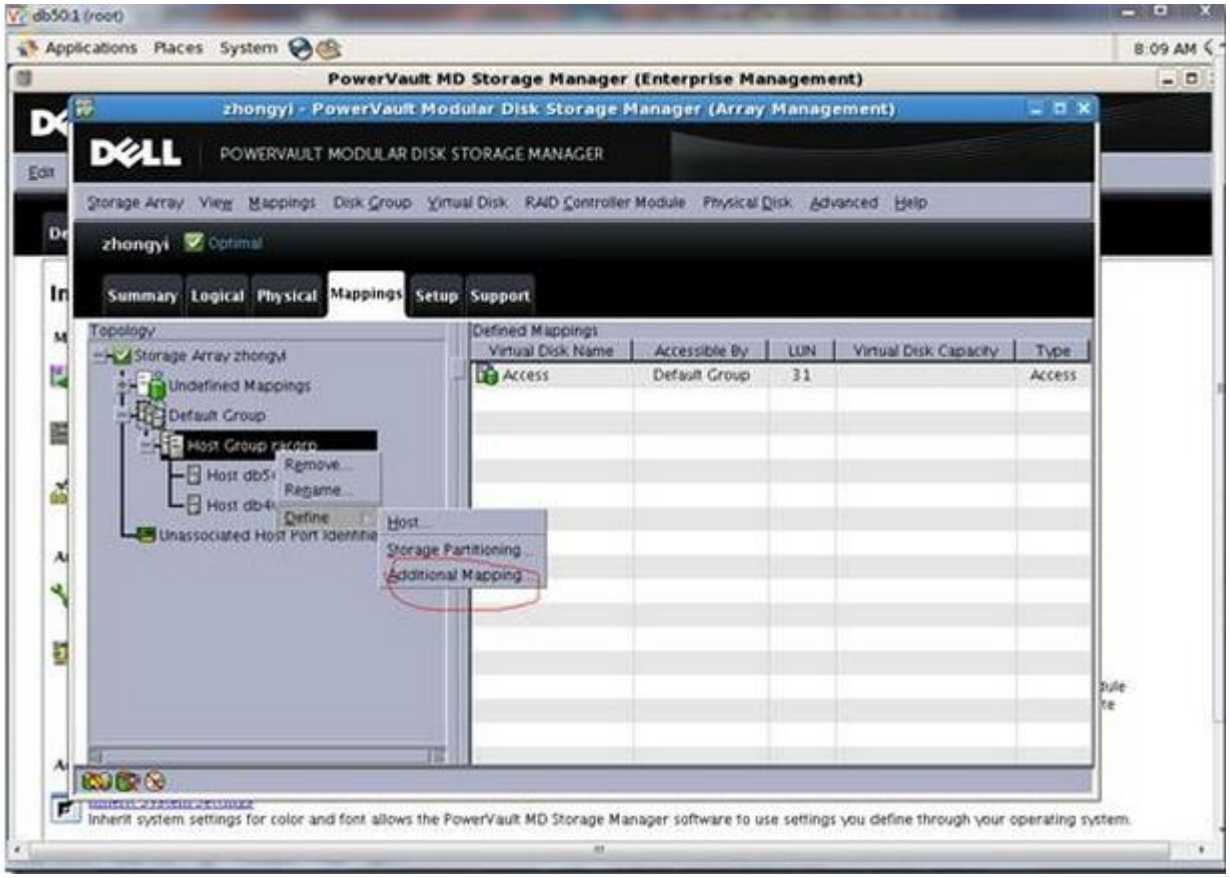


定义好第一个主机后的界面为

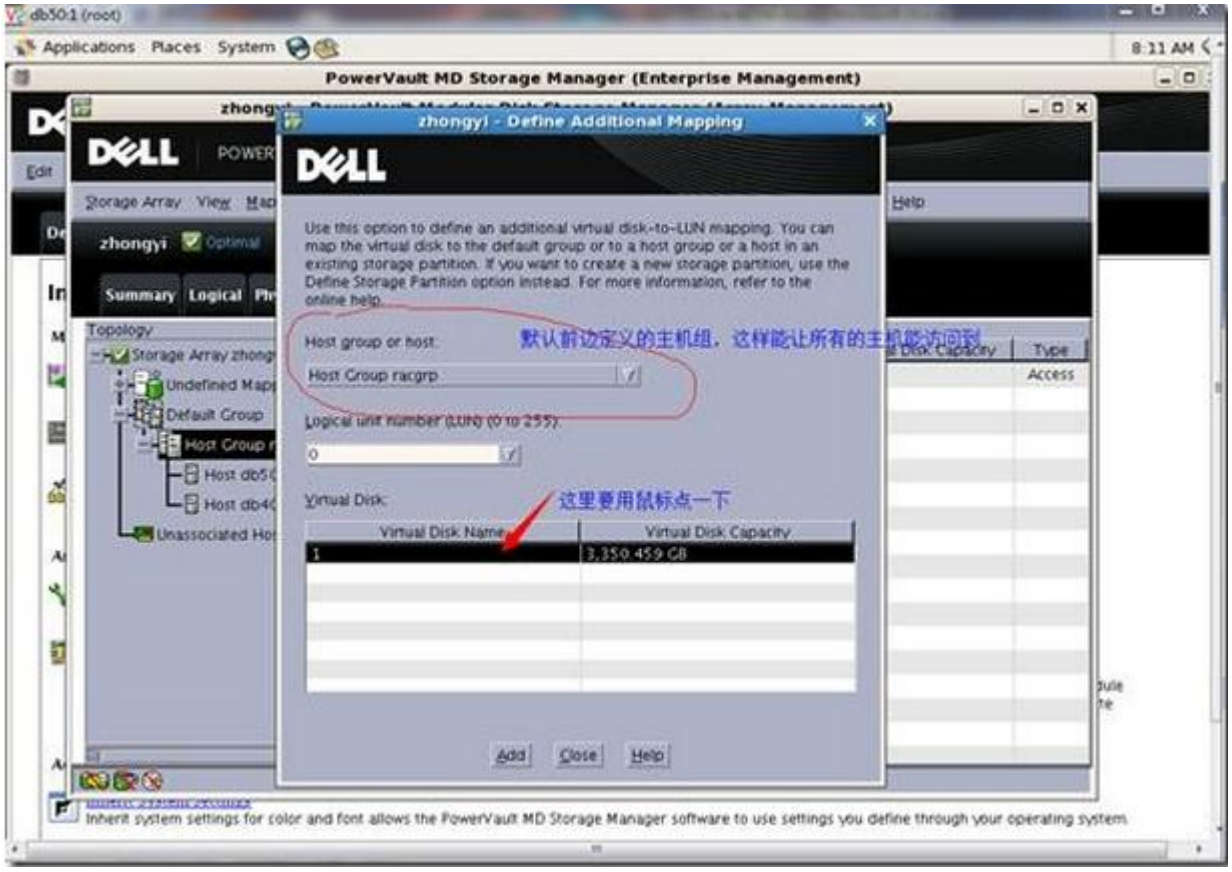
用同样的方法，把余下的主机也定义好。



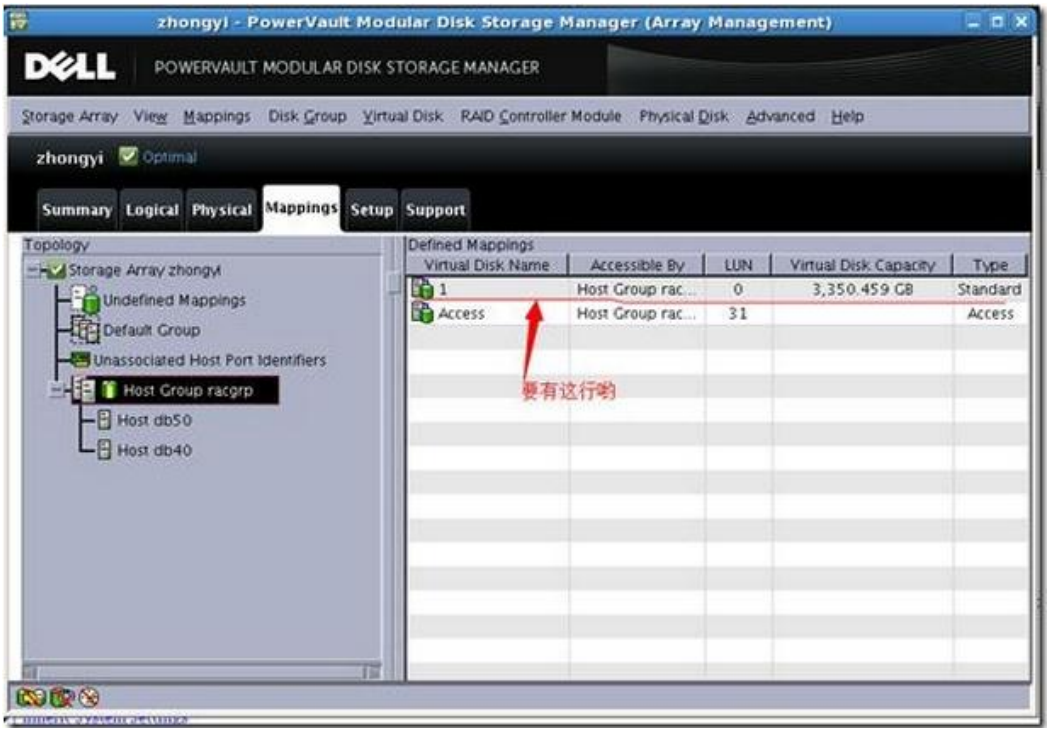
11、定义附件映射，亦即是给主机分配可访问的磁盘空间



12、点击附加映像



13、选定虚拟磁盘（前边做 raid 创建出来的），然后鼠标点击 Add



◎检查 rac 节点服务器是否识别上述步骤创建的虚拟磁盘（每个节点都要检查，确保其正确性）。在主机上执行 fdisk –l

```
[root@db50 ~]# fdisk -l

Disk /dev/sda: 256.0 GB, 256060514304 bytes
255 heads, 63 sectors/track, 31130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        1912    15358108+  83  Linux
/dev/sda2            1913        4589    21503002+  83  Linux
/dev/sda3            4590        7011    19454715   83  Linux
/dev/sda4            7012       31130   193735867+   5  Extended
/dev/sda5            7012        9306    18434556   82  Linux swap / Solaris
/dev/sda6            9307       11218    15358108+  83  Linux
/dev/sda7           11219       12238     8193118+  83  Linux

Disk /dev/sdb: 300.0 GB, 300000000000 bytes
255 heads, 63 sectors/track, 36472 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System

Disk /dev/sdc: 3597.5 GB, 3597529031168 bytes
255 heads, 63 sectors/track, 437374 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Disk /dev/sdc doesn't contain a valid partition table

Disk /dev/dm-0: 300.0 GB, 300000000000 bytes
255 heads, 63 sectors/track, 36472 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

     Device Boot      Start         End      Blocks   Id  System
[root@db50 ~]#
```

刚定义的虚拟磁盘

如果没有识别，重启一下系统就好了。

◎主机磁盘分区（仅需在一个主机进行，完成后在其它主机查看分区情况）

因为容量大于 2TB，因此需要使用 parted 进行分区。根据规划，我们需要分 6 个分区，分别用来存储 OCR、FLASH AREA 及数据文件。

◆分区前，使用 multipath –ll 查看磁盘设备文件的名称，确保分区在正确的设备上进行

```
[root@db40 mapper]# multipath -ll
mpath2 (3690b11c0001b0b9a0000037553100f24) dm-1 DELL,MD32xx
[size=3.3T][features=3 queue_if_no_path pg_init_retries 50][hwhandler=1 rdac][rw]
\_ round-robin 0 [prio=100][enabled]
 \_ 1:0:0:0 sdc 8:32  [active][ghost]
\_ round-robin 0 [prio=0][enabled]
 \_ 1:0:1:0 sdd 8:48  [active][ready]
mpath1 (35000c5006be9d36b) dm-0 SEAGATE,ST3300657SS
[size=279G][features=0][hwhandler=0][rw]
\_ round-robin 0 [prio=1][active]
 \_ 0:0:1:0 sdb 8:16  [active][ready]
[root@db40 mapper]#
```

◆进入目录 /dev/mapper，查看是否存在 mpath*这样的文件

```
[root@db40 mapper]# ll
total 0
crw------- 1 root root  10, 60 Mar  2 09:55 control
brw-rw---- 1 root disk 253,  0 Mar  2 09:55 mpath1
brw-rw---- 1 root disk 253,  1 Mar  2 10:24 mpath2
```

本例显示的磁盘阵列映射的设备名是 mpath2,那么用 dparted 分区，就要在/dev/mapper/mpath2 上进行了。分区后的结果如下

```
[root@db50 ~]# parted /dev/mapper/mpath2
GNU Parted 1.8.1
Using /dev/mapper/mpath2
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) p

Model: Linux device-mapper (dm)
Disk /dev/mapper/mpath2: 3598GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number  Start   End     Size    File system  Name     Flags
 1      17.4kB  518MB   518MB                primary
 2      518MB   1037MB  519MB                primary
 3      1038MB  209GB   208GB                primary
 4      209GB   417GB   208GB                primary
 5      417GB   1590GB  1173GB               primary
 6      1590GB  3598GB  2008GB               primary

(parted)
```

◆检查分区是否成功？进入目录/dev/mapper，可以看到分区后新生成的文件

```
[root@db40 mapper]# ll
total 0
crw------- 1 root root  10, 60 Mar  2 09:55 control
brw-rw---- 1 root disk 253,  0 Mar  2 09:55 mpath1
brw-rw---- 1 root disk 253,  1 Mar  2 10:24 mpath2
brw-rw---- 1 root disk 253,  2 Mar  2 10:24 mpath2p1
brw-rw---- 1 root disk 253,  3 Mar  2 10:24 mpath2p2
brw-rw---- 1 root disk 253,  4 Mar  2 10:24 mpath2p3
brw-rw---- 1 root disk 253,  5 Mar  2 10:24 mpath2p4
brw-rw---- 1 root disk 253,  6 Mar  2 10:24 mpath2p5
brw-rw---- 1 root disk 253,  7 Mar  2 10:24 mpath2p6
[root@db40 mapper]#
```

（三）安装和配置自动存储管理 ASM

oracle 11g rac 推荐的存储方式为 ASM（Automatic Storage Management），具体功能和概念，就没必要在这里重复。我们需要根据系统的内核版本，在官网下载与内核相一致的 rpm 安装包。

```
[root@db50 vm_dir]# uname -an
Linux db50 2.6.18-348.el5 #1 SMP Tue Jan 8 17:53:53 EST 2013 x86_64 x86_64 x86_64 GNU/Linux
[root@db50 vm_dir]#
```

一共三个，如图所示：

```
[root@db50 vm_dir]# ll *.rpm
-rw-r--r-- 1 root root 25420 Jan 16  2013 oracleasm-2.6.18-348.el5-2.0.5-1.el5.x86_64.rpm
-rw-r--r-- 1 root root 14176 Mar  2 11:25 oracleasmlib-2.0.4-1.el5.x86_64.rpm
-rw-r--r-- 1 root root 91430 Mar  2 11:25 oracleasm-support-2.1.7-1.el5.x86_64.rpm
[root@db50 vm_dir]#
```

如果下载来的包跟内核不匹配，将会导致 asm 初始化失败，不能进行进一步的操作。这是初学者很容易犯的错误，切记了！

◎安装 asm 相关的软件包

执行 rpm –ivh oracleadm*进行安装。这几个包之间有依赖关系，如果一个个单独安装，需要按某种顺序进行，如果使用通配符*，就省事多了，它能自动进行依赖处理。

```
[root@db50 vm_dir]# rpm -ivh oracle*.rpm
warning: oracleasm-2.6.18-348.el5-2.0.5-1.el5.x86_64.rpm: Header V3 DSA signature: NOKEY, key ID 1e5e0159
Preparing...                ########################################### [100%]
   1:oracleasm-support      ########################################### [ 33%]
   2:oracleasm-2.6.18-348.el########################################### [ 67%]
   3:oracleasmlib           ########################################### [100%]
[root@db50 vm_dir]# /etc/init.d/oracleasm scandisks
Scanning the system for Oracle ASMLib disks:  [  OK  ]
[root@db50 vm_dir]#
```

安装完 oracleasm 以后，至少会生成目录 /dev/oracleasm/disks，这时我们可以查看一下这个目录，发现它里面是空的，接下来的操作，就可以在目录里生成如果文件了。

◎执行/etc/init.d/oracleasm configure ，进行初始化，根据输出指定 asm 的属主及组。

◎接下来，就可以创建 asm 磁盘组了，所执行的命令如下：

```
[root@db40 ~]# service oracleasm start

Initializing the Oracle ASMLib driver: [  OK  ]

Scanning the system for Oracle ASMLib disks: [  OK  ]

[root@db40 ~]# /etc/init.d/oracleasm createdisk  OCR_VOL1

/dev/mapper/mpath2p1

Marking disk "OCR_VOL1" as an ASM disk: [  OK  ]

[root@db40 ~]# /etc/init.d/oracleasm createdisk  OCR_VOL2

/dev/mapper/mpath2p2
```

```
Marking disk "OCR_VOL2" as an ASM disk: [  OK  ]
[root@db40 ~]# /etc/init.d/oracleasm createdisk  FLASH_VOL1
/dev/mapper/mpath2p3
Marking disk "FLASH_VOL1" as an ASM disk: [  OK  ]
[root@db40 ~]# /etc/init.d/oracleasm createdisk  FLASH_VOL2
/dev/mapper/mpath2p4
Marking disk "FLASH_VOL2" as an ASM disk: [  OK  ]
[root@db40 ~]# /etc/init.d/oracleasm createdisk  DATA_VOL1
/dev/mapper/mpath2p5
Marking disk "DATA_VOL1" as an ASM disk: [  OK  ]
[root@db40 ~]# /etc/init.d/oracleasm createdisk  DATA_VOL2
/dev/mapper/mpath2p6
Marking disk "DATA_VOL2" as an ASM disk: [  OK  ]
```

我一共创建 6 个 oracle asm 磁盘，这样做的好处是把数据尽可能的分散到不同的磁盘区域，后边再把 2 个 asm 磁盘合并成一个磁盘组，提高可用性和扩充空间。

如果执行过程出现故障，可通过日志文件/var/log/oracleam 看出端倪。执行完以后，用一下方法检查是否如愿（一次安装过程中，发现一个节点没有目录/dev/oracleasm，日志/var/log/oracleasm 有“oracleasm-read-label: Unable to open device "/dev/sdb1": No such file or directory”，折腾半天，原来是没执行/etc/init.d/oracleasm enable）：

◎检查 oracleasm 创建的磁盘是否成功

（1） 查看系统目录，看是不是至少存在 6 个文件

```
[root@db40 ~]# ll /dev/oracleasm/disks/
total 0
brw-rw---- 1 grid oinstall 253, 6 Mar  2 13:44 DATA_VOL1
brw-rw---- 1 grid oinstall 253, 7 Mar  2 13:46 DATA_VOL2
brw-rw---- 1 grid oinstall 253, 4 Mar  2 13:44 FLASH_VOL1
brw-rw---- 1 grid oinstall 253, 5 Mar  2 13:44 FLASH_VOL2
brw-rw---- 1 grid oinstall 253, 2 Mar  2 13:42 OCR_VOL1
brw-rw---- 1 grid oinstall 253, 3 Mar  2 13:43 OCR_VOL2
[root@db40 ~]#
```

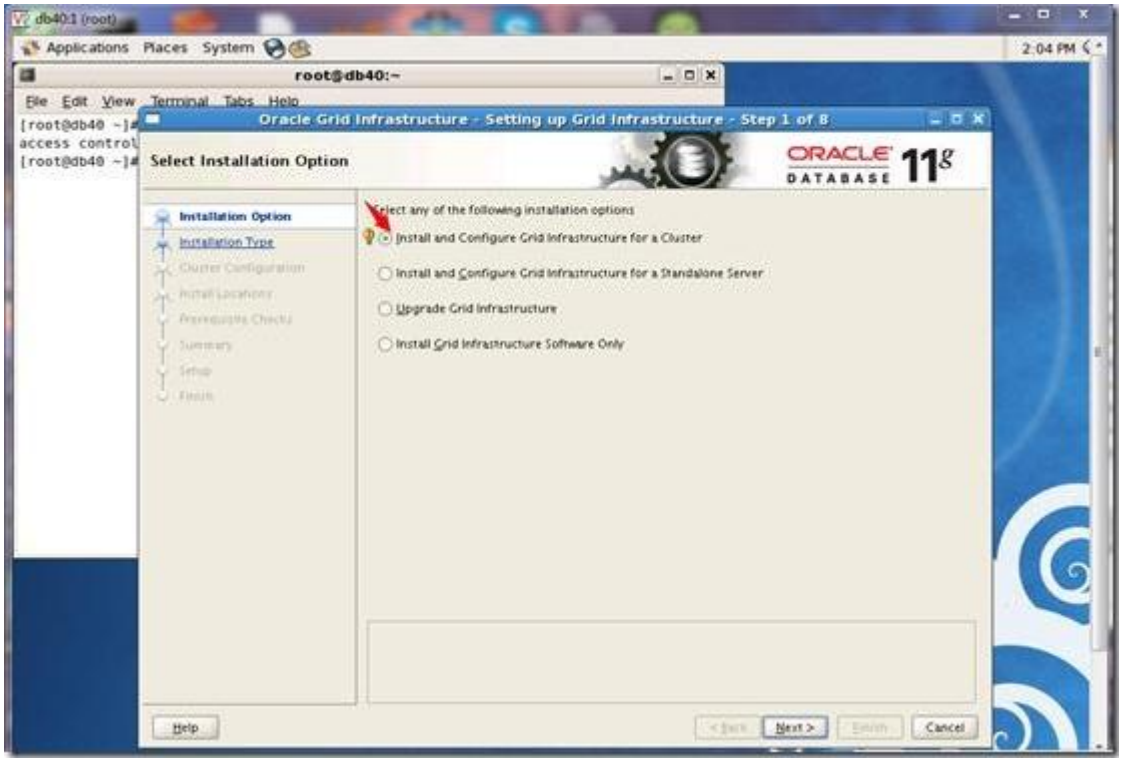
（2）从逻辑上查看 asm 磁盘组，记得要在每一个 rac 主机查询

```
[root@db50 ~]# /etc/init.d/oracleasm scandisks

Scanning the system for Oracle ASMLib disks: [  OK  ]

[root@db50 ~]# /etc/init.d/oracleasm listdisks

DATA_VOL1

DATA_VOL2

FLASH_VOL1

FLASH_VOL2

OCR_VOL1

OCR_VOL2
```

（四）安装集群软件 grid

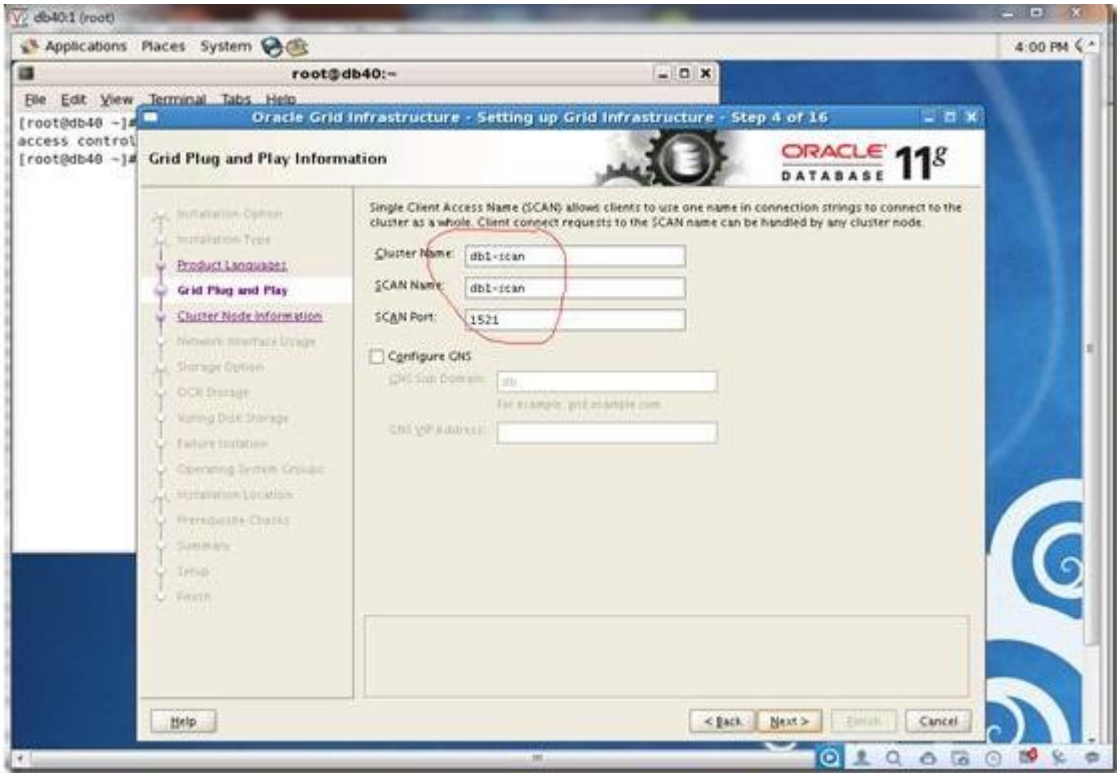
◎grid 软件安装（安装过程只需在一个节点执行，有脚本需要同时在每个节点都运行，请注意看安装提示就可以了）。

Vnc 连接服务器，打开终端，执行 xhost + ，接着进入源安装文件目录，执行命令 ./runInstaller，如果正常，将弹出如下图的安装界面：

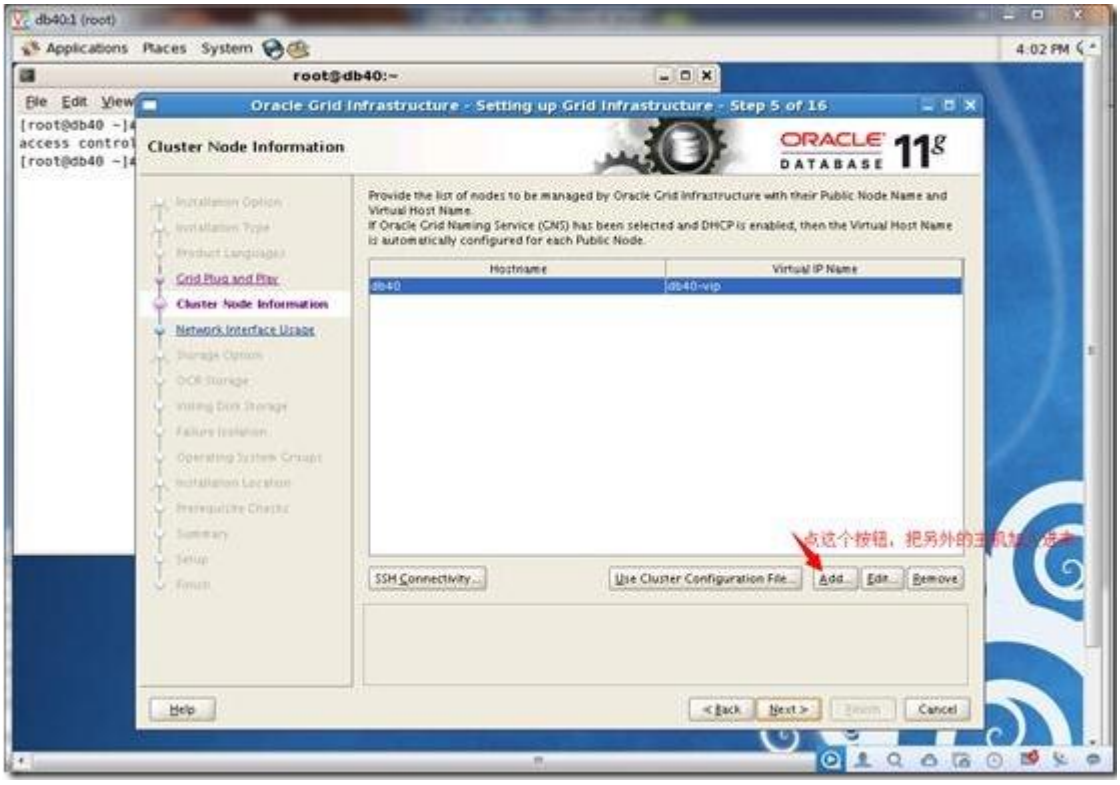


选择典型安装

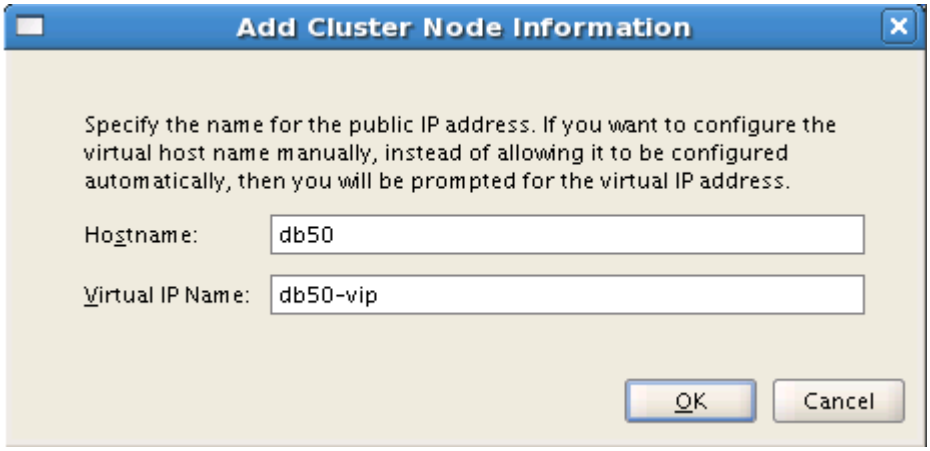
SCAN 的名称，要与/etc/hosts 完全一致，**特别注意，在同一个网段内，要保证这个命名的唯一性。**如果在同一网段内部署多个 oracle rac 集群，scan 的命名如果重复了，安装将不能继续进行！！！



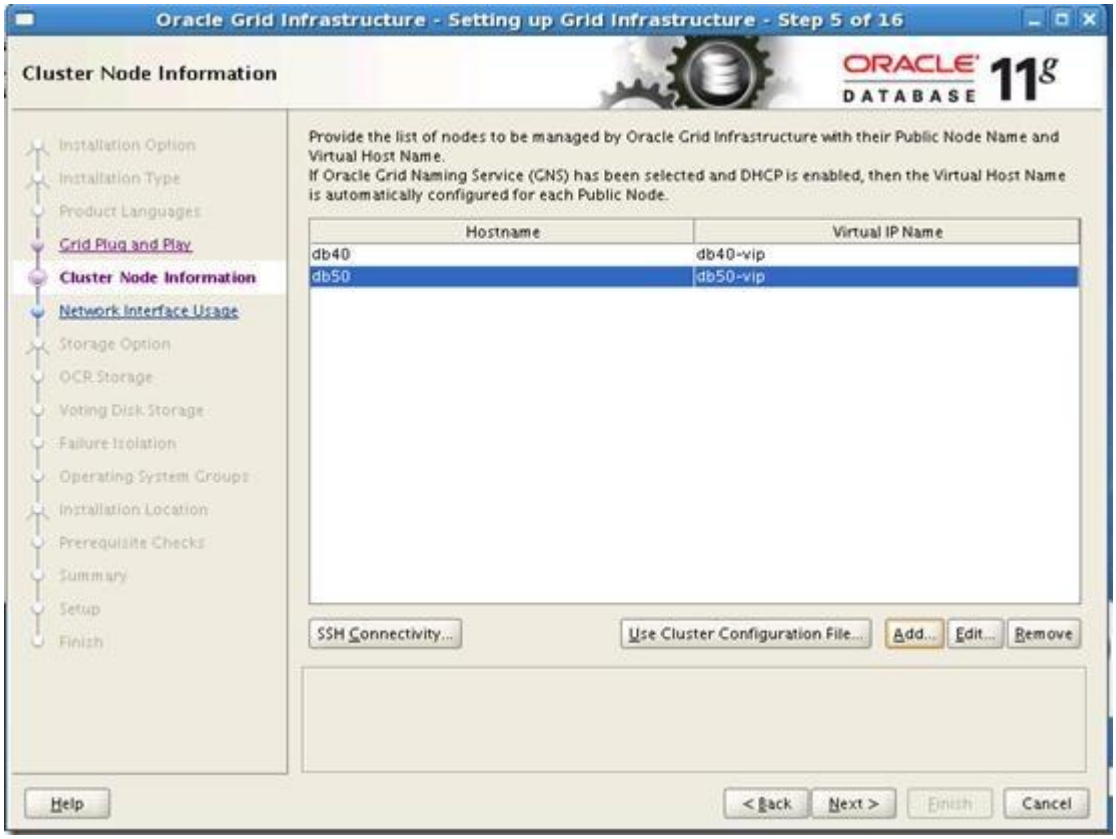
加入余下的主机（默认情况下，只有一个主机显示）



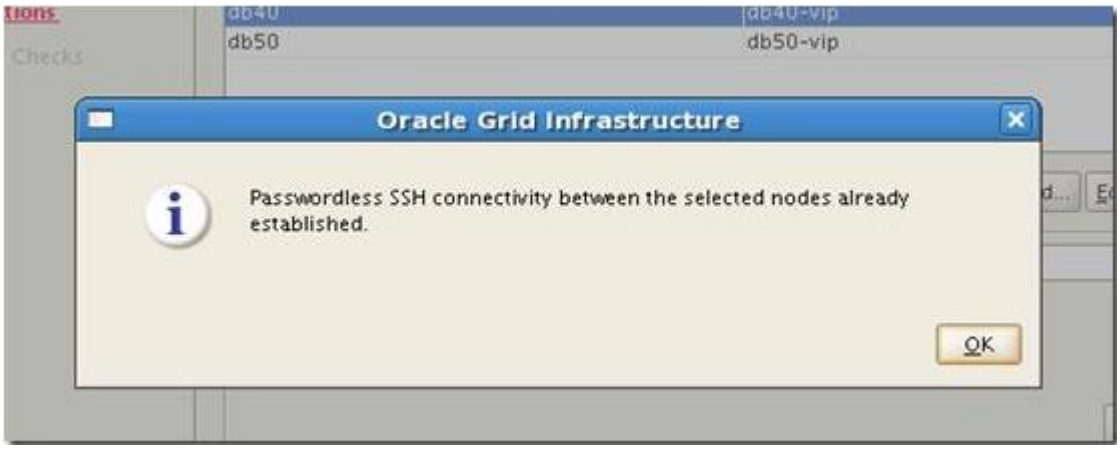
一定要对着/etc/hosts 定义好的填写



添加好后，主机名应该显示在列表中

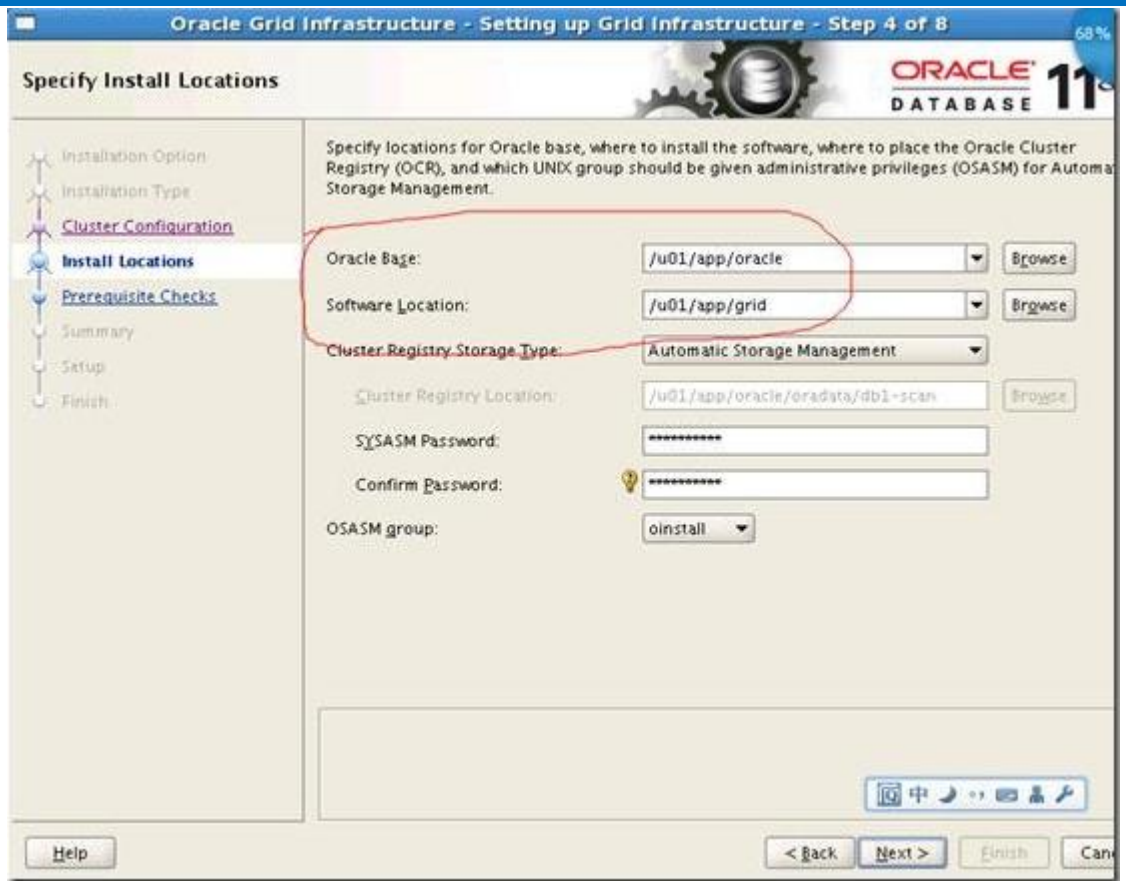


Ssh 连接验证，点按钮”SSH Connectivity …”，这里一定要做好 ssh 无密码验证，否则下一步不能进行（我在这里费了好长的时间啊）。

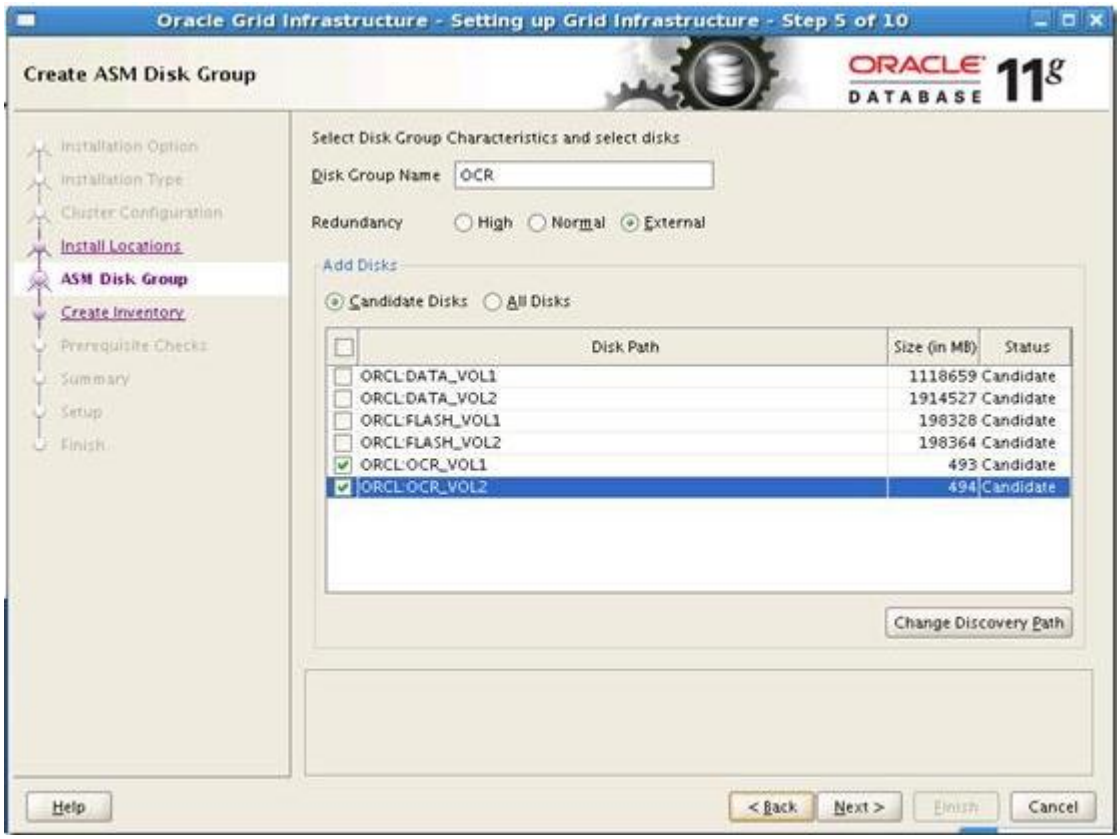


说明：ssh 连接，也可以不用手工设置，直接点那个 setup 进行处理，省事不少。

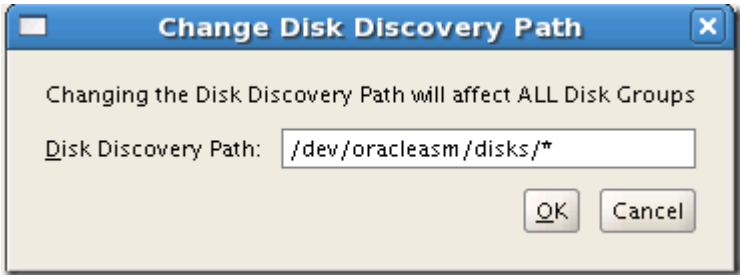
选择安装目录及存储位置（注意 base 要有 location 不一样哟）



创建 asm 磁盘组，因为磁盘阵列做了 raid10，因此下边的冗余等级就选择外部的（External）

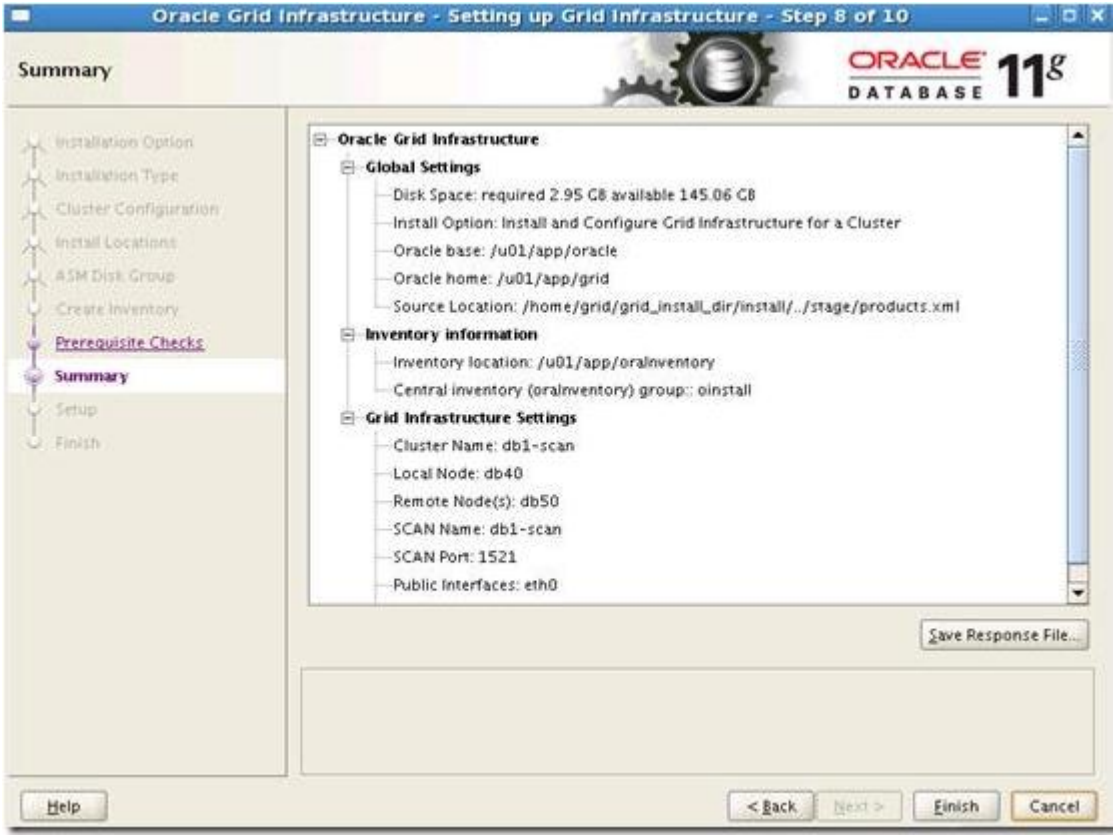


这些名字看着眼熟吧？就是前边 oracleasm createdisk 创建出来的。这里勾选 2 个项目，目的在于把他们合并成一个 asm 磁盘组。如果此处没有找到候选磁盘，则需点击按钮“Change Discovery Path”就，进行手工查找。



最好输入全路径，幸运的是，这里支持通配符.这里创建的磁盘组，是用来存储 asm 磁盘仲裁等信息。

再往下进行校验，突然弹出一个报错信息，是关于 ntp 的。先关闭 ntpd 服务，然后删掉配置文件 /etc/ntp.conf,这样就排除了故障。



到这里以后，一般不会出什么状况，只管点按钮“完成”进行文件的提取和安装，执行到一定程度后，需要以 root 帐号执行两个脚本/u01/app/oraInventory/orainstRoot.sh、/u01/app/grid/root.sh，后一个脚本很费时间，也容易出错。

◆第一个服务器，执行一次通过，但第二个服务器执行时，出现错误信息：

```
ASM failed to start. Check /u01/app/oracle/cfgtoollogs/asmca/asmca-
14030210PM2304.log for details.


Configuration of ASM failed, see logs for details
Did not succssfully configure and start ASM
CRS-2500: Cannot stop resource 'ora.crsd' as it is not running
CRS-4000: Command Stop failed, or completed with errors.
Command return code of 1 (256) from command: /u01/app/grid/bin/crsctl
stop resource ora.crsd -init
Stop of resource "ora.crsd -init" failed
```

通过对比检查，发现两个服务器的目录属主有不一致的地方。正确运行脚本成功的那个/dev/oracleasm/下的目录属主是 grid:oinstall，而有问题的那个主机的属主是 root:root，执行/ete/init.d/oracleasm 指定属主和组，保证文件的属组和主是正确的，然后执行./roothas.pl -delete -force -verbose ，删除不成功的配置，最后执行/u01/app/grid/root.sh 完成安装。注意：安装界面有 2 个失败，是因为私有地址做的 public,可以无视。

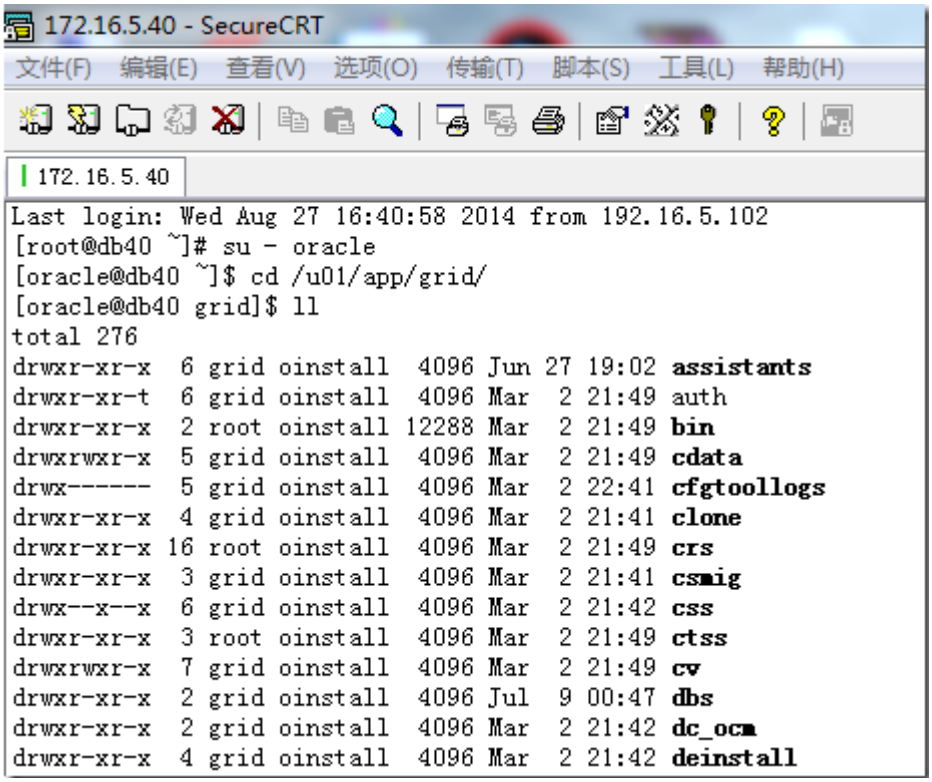
◎检查安装的正确性

（1） 查看 asm 进程

（2） 以 grid 帐号，执行 asmcmd ，提示符下 ls 查看目录或者文件，有内容者则是我们锁需要的。

```
[grid@db50 ~]$ asmcmd
ASMCMD> ls
OCR/
ASMCMD> cd ocr
ASMCMD> ls
db1-scan/
ASMCMD> cd db1-scan
ASMCMD> ls
ASMPARAMETERFILE/
OCRFILE/
ASMCMD>
```

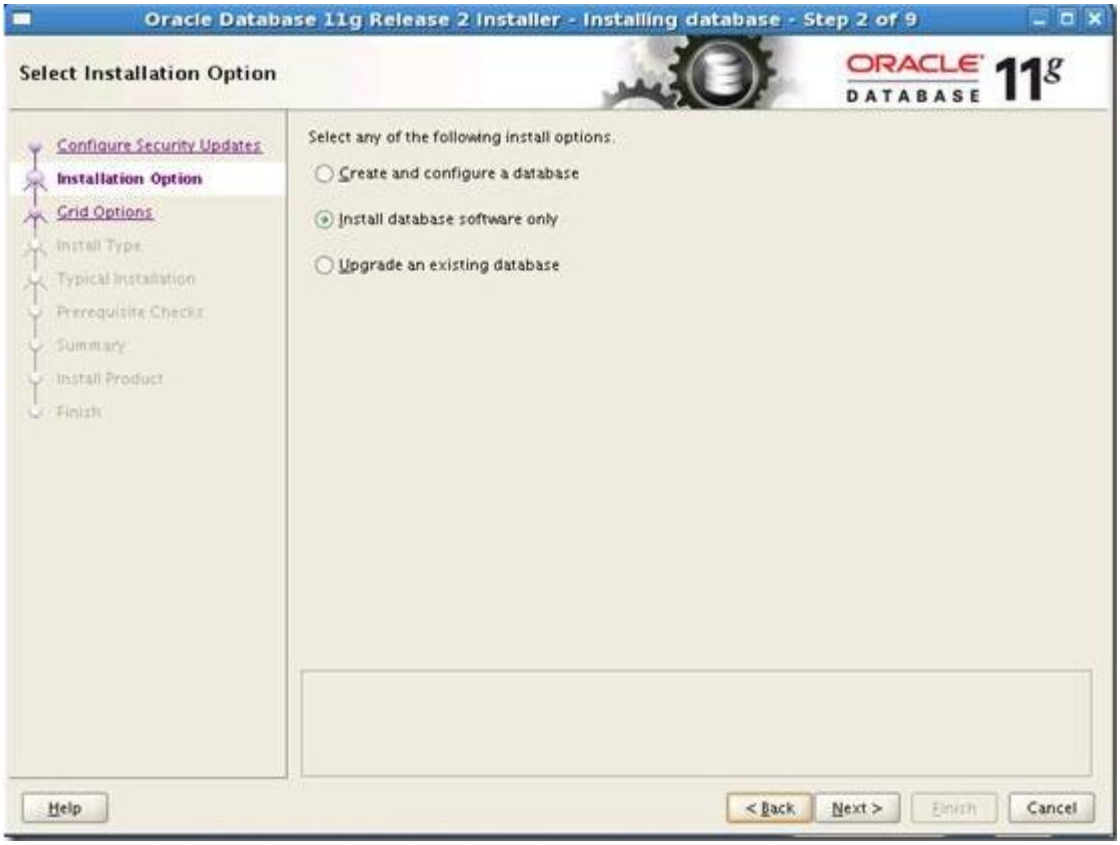
（3） 查看安装目录，看是否生成相关的目录很文件



（五）Oracle 安装和数据库创建

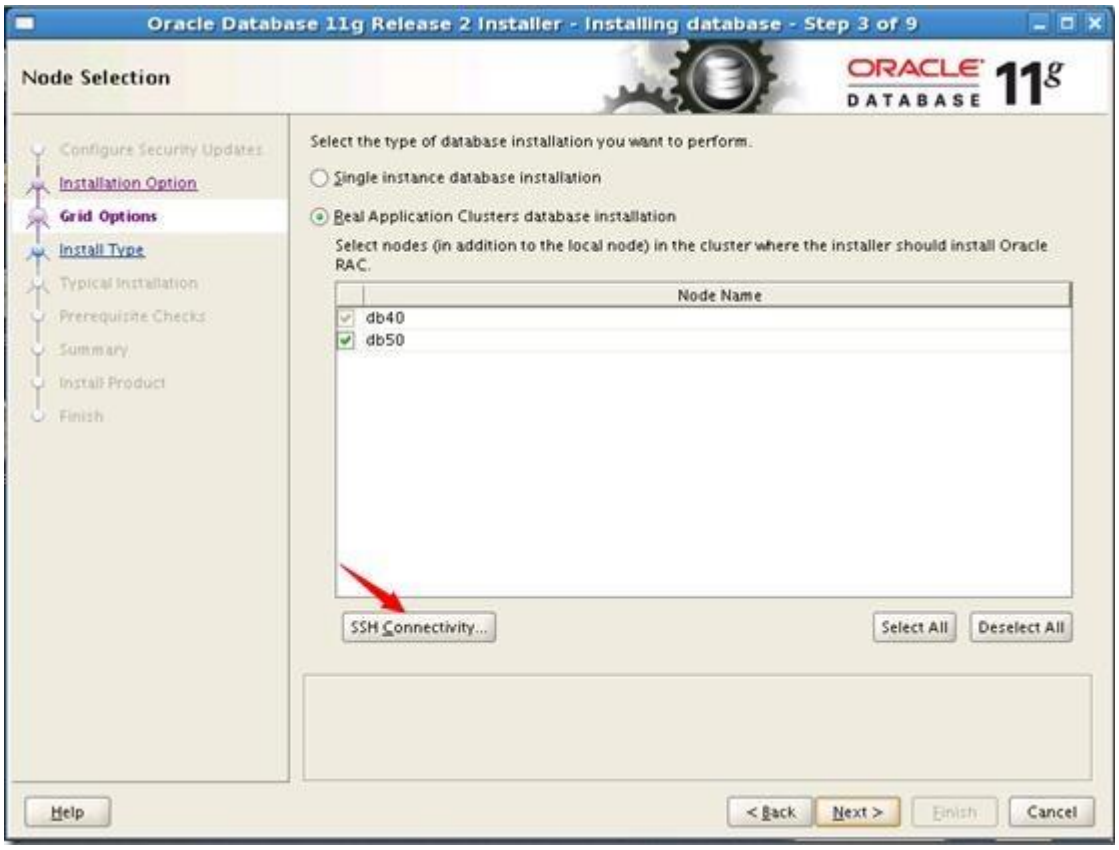
◎安装数据库软件（在一个节点进行操作，安装过程中，有脚本需要同时在每个节点都运行，请注意看安装提示就可以了）

与安装 grid 一样，还是需要用 vnc 连接到某个节点的系统，执行 xhost + ,然后再打开一个终端，执行 su – oracle 切换到 oracle 用户。再进入安装源文件的目录，执行脚本./runInstaller，如果正常，将弹出如下图的安装界面：



先安装软件，后边单独创建数据库。

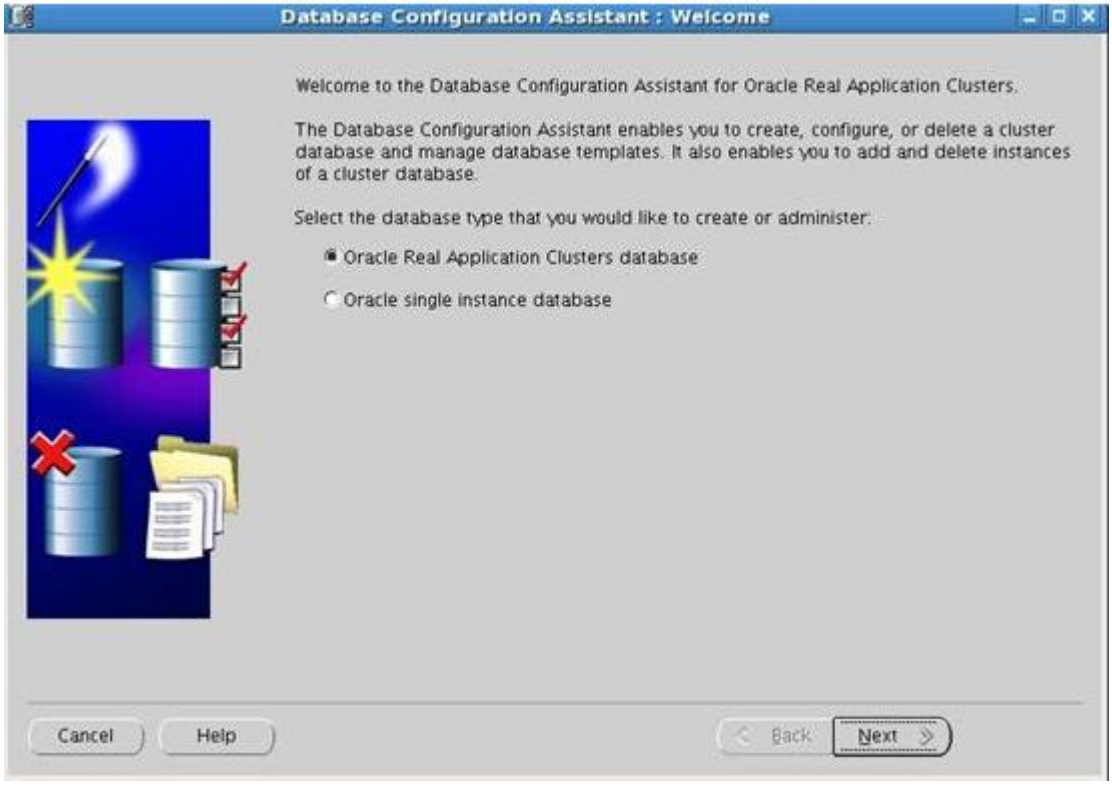
跟 grid 一样，需要先进行 ssh 验证



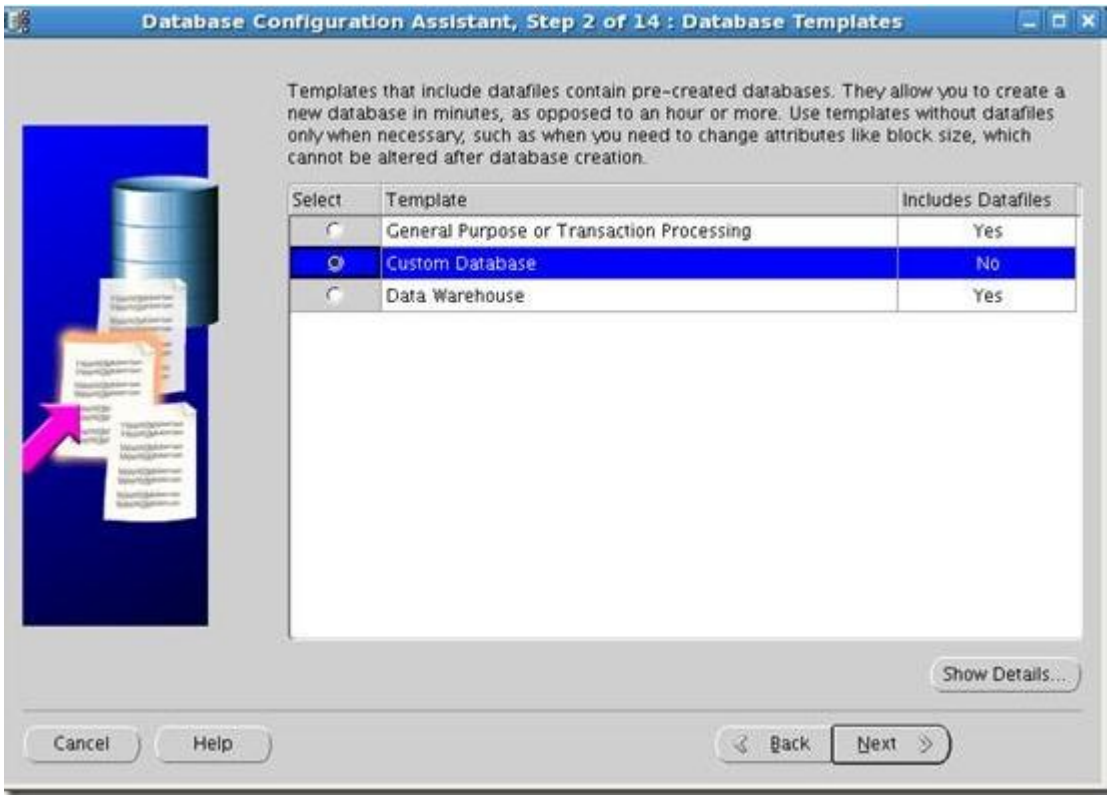
验证成功以后，后边的步骤不容易出错，也跟单实例安装基本相同。

◎创建数据库（在一个节点上操作）

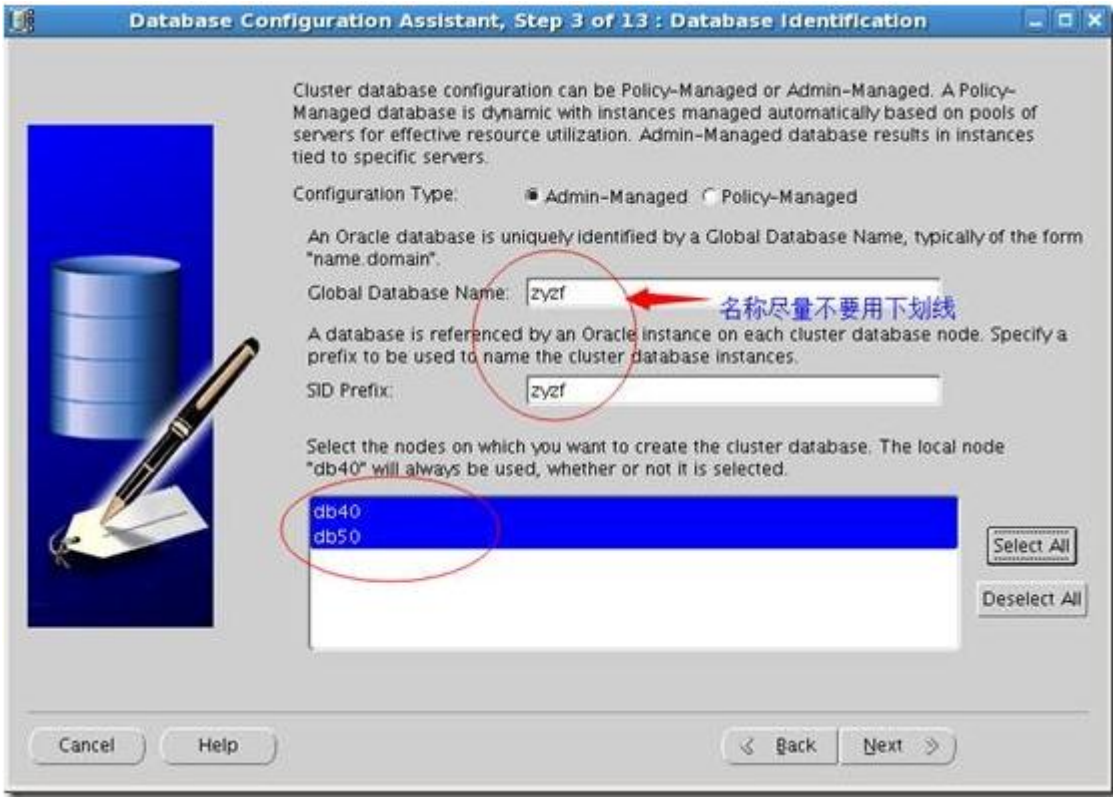
以 oracle 帐号执行 dbca



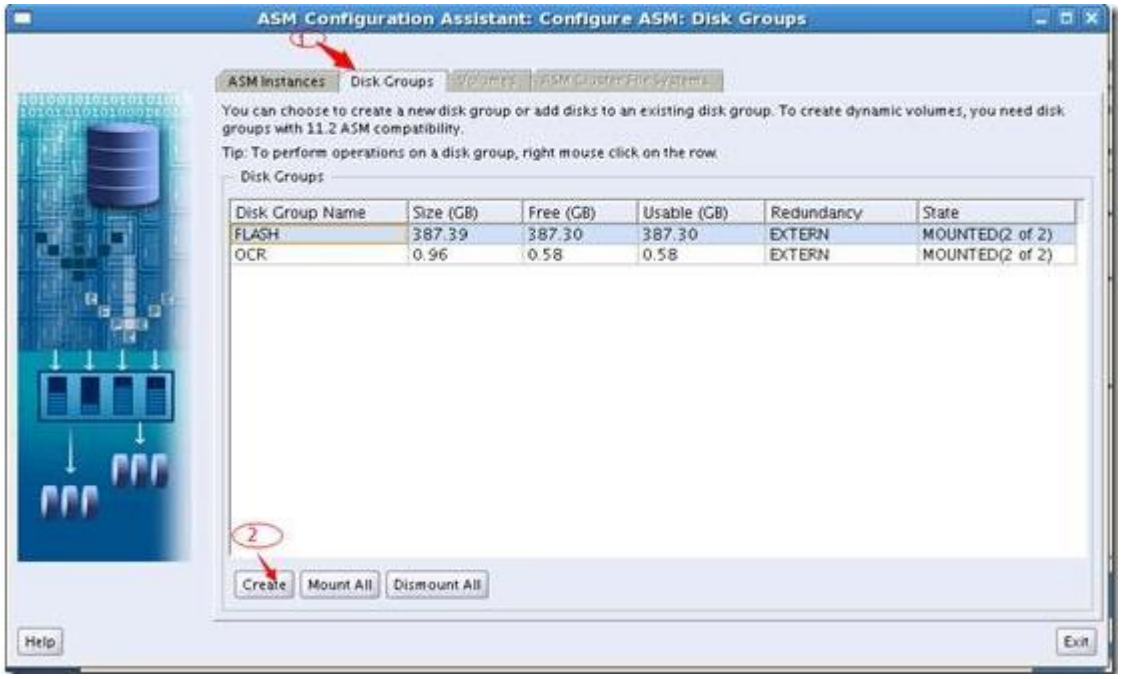
若干下一步后，咱们还是选择定制方式吧（Custom Database）



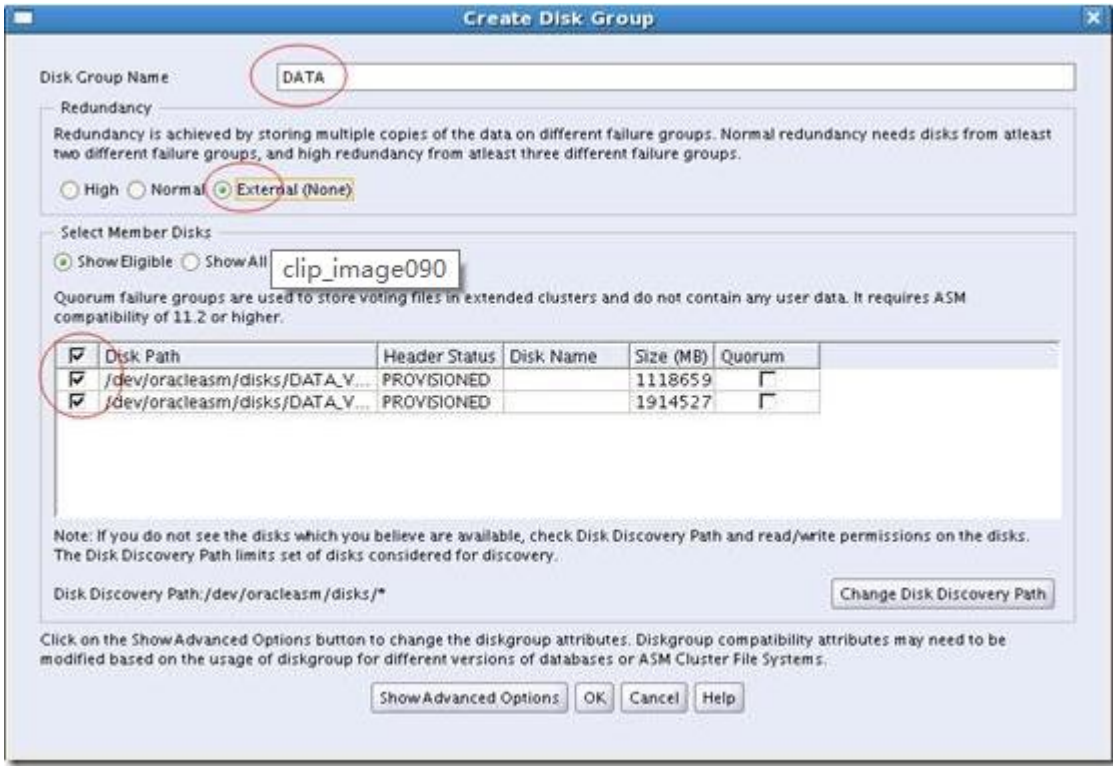
记得把所有的主机都选上



创建所需 asm 磁盘组（前边安装 grid 时，已经创建过一组了）：grid 用户执行 asmca

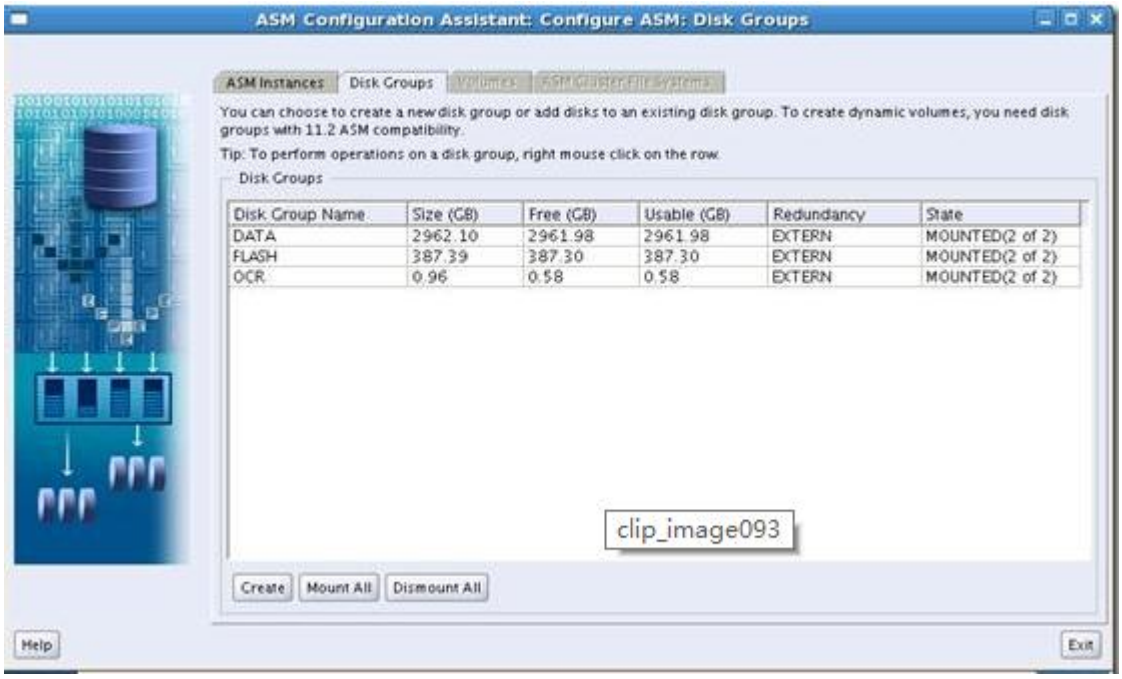


手工输入和选定一些项目

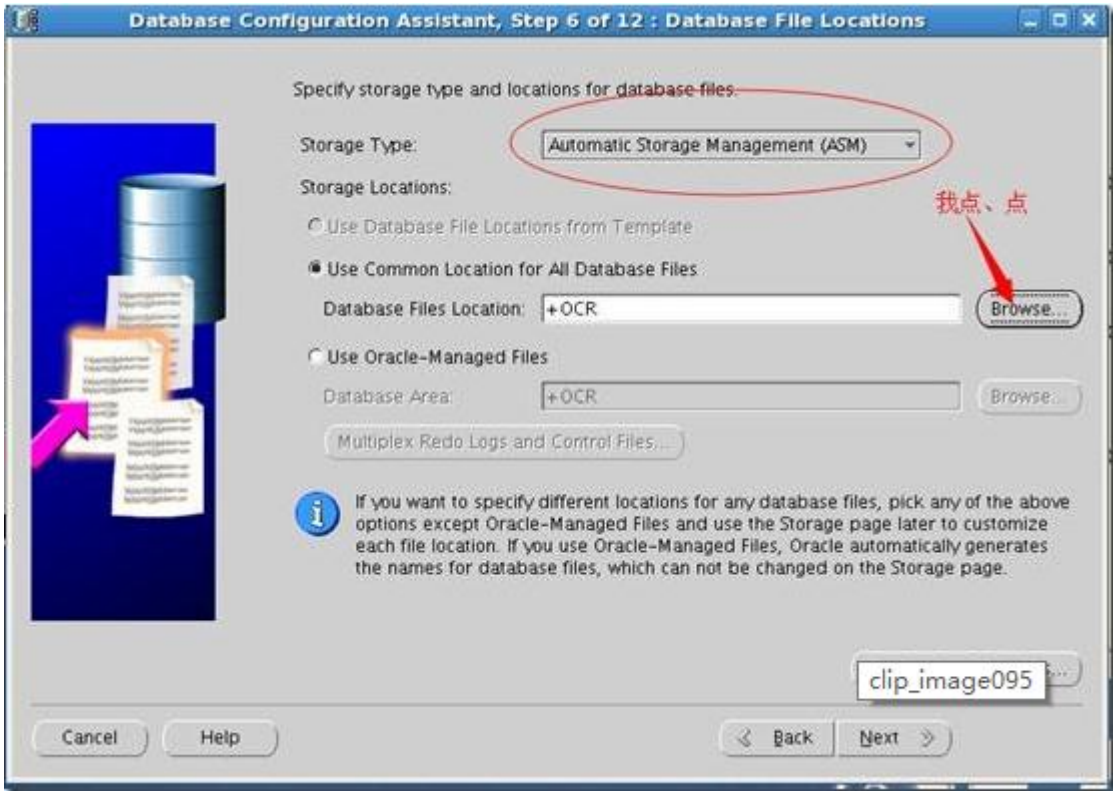


转呀转，要一点点耐心，嘿！刚敲到这里，就出来了

创建好后，目前还可以改哟。将来数据库上线运行了，不能再做这个尝试了。现有环境创建完后汇总输出如下



选择数据库文件存放的位置。当然要选 ASM。如果不执行这个磁盘组创建过程，后面的数据库文件存放位置，如果选 asm，将不能往下进行。



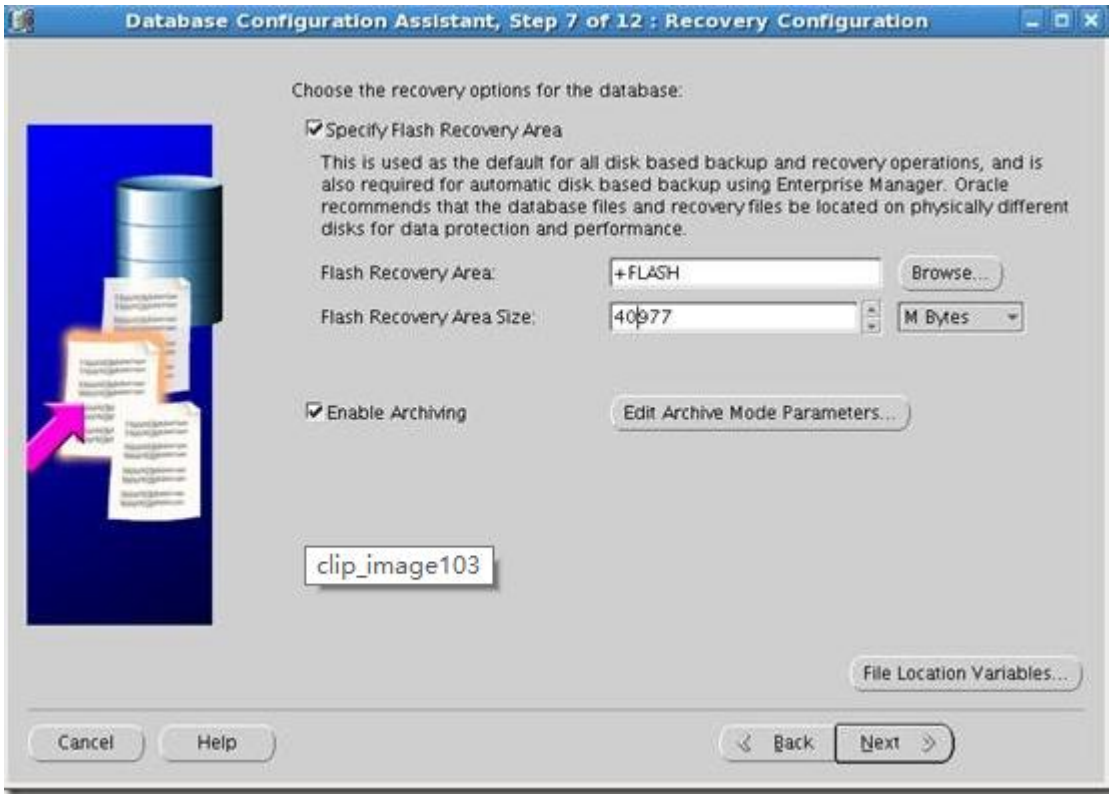
选容量最大的这个（OCR 已经被前边的操作占用了，FLASH 留到后边用）

往下一步的时候，需要输入 asm 的一个密码。密码在前天的 grid 安装设定的哟，要记牢。

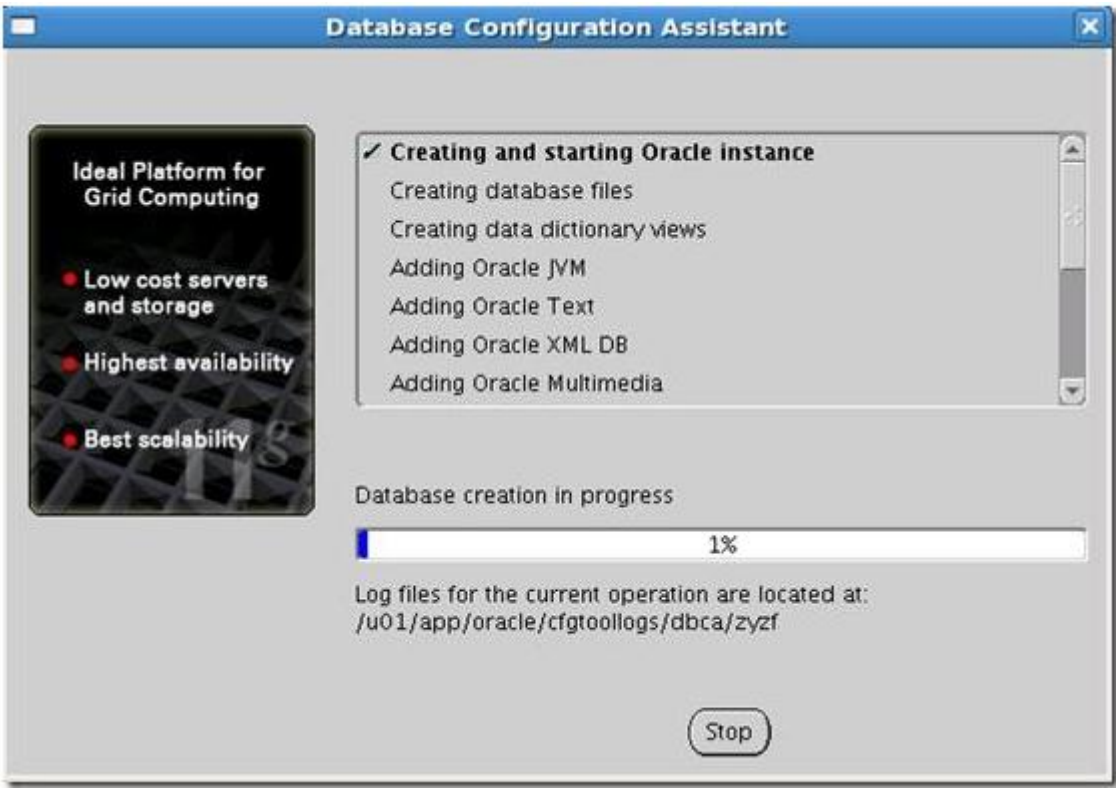
归档区也用 asm，这就用上前边剩下的 asm 磁盘组 FLASH，一个也没多余。

自己动手，选大的

归档要打勾哟（当然实例运行起来后，也是可以打开的）



后边几步，就不用截图了。到这里，就慢慢等着吧



部署过程，很漫长，也很考验人的耐性。

后续及注意事项

◆oracle rac 数据库创建后，默认的一些系统表空间太小，运行不了多久，就会达到 90%以上，最好在创建完毕后把它改大一些。

◆oracle rac 监听器监听的 ip 地址是虚拟地址，如果尝试连接实际物理地址，将永远不会得逞。

◆oracle rac 心跳检查导致大量的数据通讯，因此对网络的质量和速度要求较高。一个运行中的 oracle rac 集群，曾因为少买了一根网线而使用机房提供的劣质六类线，导致心跳检测时好时坏，引起实例关闭。

◆用户表空间，固定大小，扩充容量以创建多个数据文件为佳。

◆默认的重做日志文件也很小，需要多创建几个日志组，并增大其容量。

◆oracle rac 运行起来后，一定要做好备份，熟练的用好 rman，将会有很多好处。

◆如果条件运行，跨机房的 dataguard 给弄上哟。

◆多路径没配好，可能会导致数据路径交替切换，一会走这条数据线，一会又走另一条数据线。直接的表现就是数据访问缓慢。通过查询报警日志可以得到有用的信息。

◆报警日志文件非常有用，要经常关注它！

◆修改/etc/sysconfig/ntp，增加行 OPTIONS="-x -u ntp:ntp -p /var/run/ntpd.pid" 重启 ntpd 服务

◆Ssh 等效要双向，即从任何一个服务器可以无密码登录另一个服务器系统

◆/etc/hosts 的第一行，要把主机名去掉；同时/etc/sysconf/network 里面的名称，要与/etc/hosts 一致

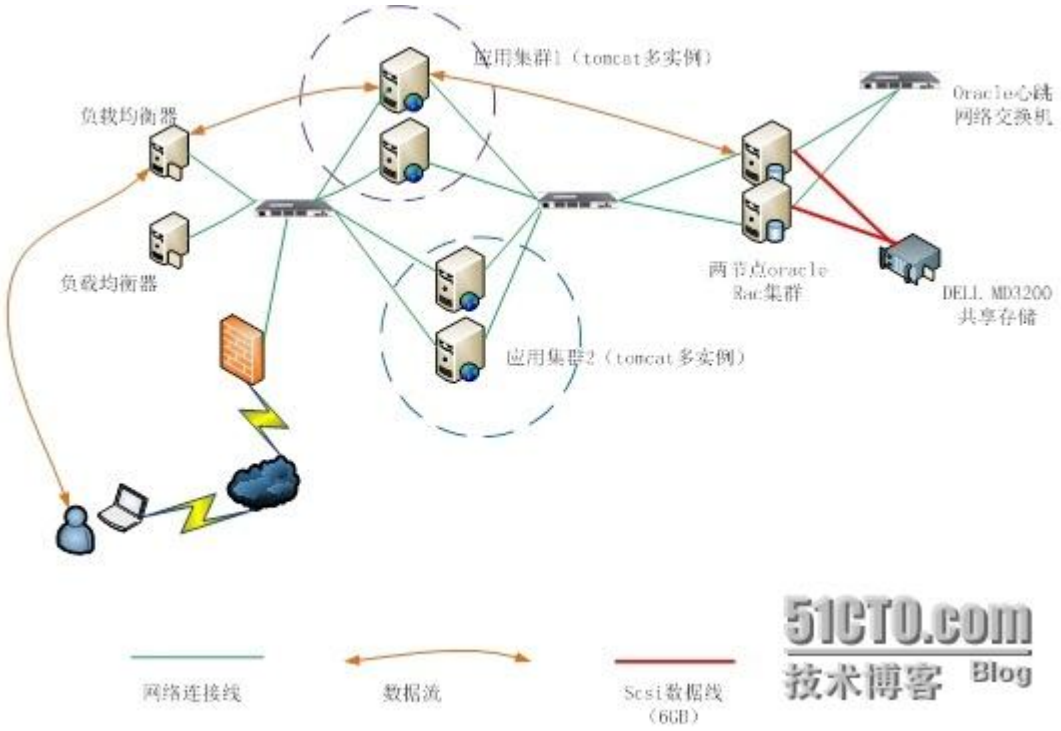
◆Oracleasm 配置时，一定要注意属主和组名。否则会有权限问题。Oracleasm configure –i 指定 interfere 为 grid,组 groups 为 oinstall。因为 oracle 帐号也属于 oinstall 组

写这些文字，花了不少时间，希望能对需要的人有所帮助。网上有一些文档，但大部分都是虚拟环境，领会不了 asm 真实场景的特性。

一次支付平台紧急故障处理备忘

作者：田逸 来源：http://sery.blog.51cto.com/10037/1432066

监控没报警直接收到故障电话，说业务全部挂了。时间紧迫，需要快速解决。



根据拓扑结构，顺序进行如下操作：

◎检查负载均衡器

负载均衡器安装 keepalived+ haproxy，先从监控界面检查运行状态，其输出如下图所示

由图可知，还有一个应用处于正常状态，因此可以大致判定负载均衡应该是正常的。

◎检查应用服务器

应用服务器由 4 个服务器组成 2 组独立的集群，每组服务器安装的软件和配置完全一样。因此，每组服务器只需要检查其中的一个服务器就可以了。登录系统，检查如下项目：

1、 检查进程，查看 tomcat 是否还在运行，执行指令 ps auxww | grep java ，两个 java 进程运行得好好的呢！

2、 检查网络状态，分别执行 netstat –anp | grep EST ，也看不出有什么异常。

3、 检查 tomcat 日志，发现一段可疑输出，片段截取如下：

```
Could not open JDBC Connection for  transaction; nested exception is
java.sql.SQLException: An attempt by a  client to checkout a Connection has
timed out.
```

问了其他技术人员，回答说今天没有做任何程序方面的修改，由此可以简单断定，可能是数据库出了问题。顺手在应用服务上测试一下数据库服务器的网络联通性，执行命令 ping 172.16.5.40,正常；再执行 telnet 172.16.5.41 1521 有正常的输出，这可以确定数据库的监听也是启动的。注意：oracle rac

监听地址是安装过程中设定的 vip，而不是实际物理接口地址，这就是什么啥 ping 的地址是

172.16.5.40，而 telnet 跟的地址是 172.16.5.41 的原因。

4、 重启一下 tomcat，故障依旧。

5、 检查系统日志，无可以信息发现。

6、 直接在浏览器输入应用服务器的可访问 url，异常。

◎检查数据库服务器

◆系统方面的检查

1、检查 oracle 相关进程，ps aux ，其输出片段为


```
grid      4947  0.0  0.1 493200 34360 ?        Ss   Jun27   0:36 asm_lmd0_+ASM2
grid      4979  0.0  0.1 493200 35244 ?        Ss   Jun27   0:00 asm_lms0_+ASM2
grid      4983  0.0  0.0 478352 14288 ?        Ss   Jun27   0:00 asm_lmhb_+ASM2
grid      4985  0.0  0.0 478352 14416 ?        Ss   Jun27   0:00 asm_mman_+ASM2
grid      4987  0.0  0.0 484188 24740 ?        Ss   Jun27   0:00 asm_dbw0_+ASM2
grid      4989  0.0  0.0 482828 21012 ?        Ss   Jun27   0:00 asm_lgwr_+ASM2
grid      4991  0.0  0.0 478352 16928 ?        Ss   Jun27   0:00 asm_ckpt_+ASM2
grid      4993  0.0  0.0 478352 14320 ?        Ss   Jun27   0:00 asm_smon_+ASM2
grid      4995  0.0  0.0 485220 27152 ?        Ss   Jun27   0:34 asm_rbal_+ASM2
grid      4997  0.0  0.0 482764 19620 ?        Ss   Jun27   0:00 asm_gmon_+ASM2
grid      4999  0.0  0.0 478352 15108 ?        Ss   Jun27   0:00 asm_mmon_+ASM2
grid      5001  0.0  0.0 478352 17400 ?        Ss   Jun27   0:00 asm_mmnl_+ASM2
grid      5003  0.0  0.0  85208 11444 ?        S    Jun27   0:00 /u01/app/grid/bin/oclskd.bin
grid      5007  0.0  0.0 479416 18268 ?        Ss   Jun27   0:00 asm_lck0_+ASM2
root      5015  0.0  0.1 774276 53040 ?        Ssl  Jun27   0:05 /u01/app/grid/bin/crsd.bin reboot
grid      5025  0.0  0.0 483948 25904 ?        Ss   Jun27   0:00 oracle+ASM2_ocr (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
grid      5027  0.0  0.0 479756 16976 ?        Ss   Jun27   0:00 asm_azmb_+ASM2
grid      5029  0.0  0.0 479980 18680 ?        Ss   Jun27   0:00 oracle+ASM2_azmb_+asm2 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=b
eq)))
root      5051  0.0  0.0  85200 11464 ?        S    Jun27   0:00 /u01/app/grid/bin/oclskd.bin
grid      5094  0.0  0.0  90860 14036 ?        S    Jun27   0:00 /u01/app/grid/bin/evmlogger.bin -o /u01/app/grid/evm/log/evmlogger.
info -l /u01/app/grid/evm/log/evmlogger.log
grid      5138  0.0  0.0 904532 32724 ?        Ssl  Jun27   1:18 /u01/app/grid/bin/oraagent.bin
root      5142  0.2  0.0 601476 18180 ?        Ssl  Jun27   5:06 /u01/app/grid/bin/orarootagent.bin
grid      5270  0.0  0.2 1539012 75844 ?       Sl   Jun27   1:56 /u01/app/grid/jdk/jre//bin/java -Doracle.supercluster.cluster.serve
r=eonsd -Djava.net.preferIPv4Stack=true -Djava.util.logging.config.file=/u01/app/grid/srvm/admin/logging.properties -classpath /u01/
app/grid/jdk/jre//lib/rt.jar:/u01/app/grid/jlib/srvm.jar:/u01/app/grid/jlib/srvmhas.jar:/u01/app/grid/jlib/supercluster.jar:/u01/app
/grid/jlib/supercluster-common.jar:/u01/app/grid/ons/lib/ons.jar oracle.supercluster.impl.cluster.EONSServerImpl
grid      5349  0.0  0.0  58040  6068 ?        Sl   Jun27   0:00 /u01/app/grid/opmn/bin/ons -d
grid      5350  0.0  0.0  63596  8380 ?        Sl   Jun27   0:00 /u01/app/grid/opmn/bin/ons -d
grid      5386  0.0  0.0 146964 13868 ?        Ssl  Jun27   0:06 /u01/app/grid/bin/tnslsnr LISTENER -inherit
oracle    5419  0.1  0.0 737616 24308 ?        Ssl  Jun27   3:51 /u01/app/grid/bin/oraagent.bin
oracle    5498  0.0  0.1 13455192 60056 ?      Ss   Jun27   0:05 ora_pmon_zyzf2
oracle    5500  0.0  0.0 13452808 16280 ?      Ss   Jun27   0:00 ora_vktm_zyzf2
oracle    5504  0.0  0.0 13454064 18112 ?      Ss   Jun27   0:00 ora_gen0_zyzf2
oracle    5506  0.0  0.0 13458944 24236 ?      Ss   Jun27   0:00 ora_diag_zyzf2
oracle    5508  0.0  0.1 13454372 57788 ?      Ss   Jun27   0:00 ora_dbrm_zyzf2
oracle    5510  0.0  0.0 13452808 16948 ?      Ss   Jun27   0:02 ora_ping_zyzf2
oracle    5512  0.0  0.0 13452808 17356 ?      Ss   Jun27   0:00 ora_psp0_zyzf2
oracle    5514  0.0  0.0 13452808 16372 ?      Ss   Jun27   0:00 ora_acms_zyzf2
```

相关进程都处于运营状态。

◆检查监听器

```
[oracle@db40 ~]$  lsnrctl status

LSNRCTL for  Linux: Version 11.2.0.1.0 - Production on 29-6 月 -2014 10:02:49
```

```
Copyright (c)  1991, 2009, Oracle.  All rights  reserved.


Connecting to  (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521))
STATUS of the  LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version  11.2.0.1.0 - Production
Start Date                09-5 月 -2014 17:42:24
Uptime                    50 days 16 hr. 20 min. 33  sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener  Parameter File    /u01/app/grid/network/admin/listener.ora
Listener Log  File          /u01/app/oracle/diag/tnslsnr/db40/listener/alert/log.xml
Listening  Endpoints Summary...
   (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=LISTENER)))
   (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=127.0.0.1)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=172.16.5.41)(PORT=1521)))
Services  Summary...
Service  "+ASM" has 1 instance(s).
  Instance "+ASM1", status READY,  has 1 handler(s) for this service...
Service  "zyzf" has 1 instance(s).
  Instance "zyzf1", status READY,  has 1 handler(s) for this service...
```

```
Service   "zyzfXDB" has 1 instance(s).

  Instance "zyzf1", status READY,   has 1 handler(s) for this service...

The command   completed successfully
```

ASM 实例及服务在监听器里注册状态都是正常的。

◆检查 asm

执行如下指令进行简单判断

```
[root@db50 ~]# su - grid

[grid@db50 ~]$ asmcmd

ASMCMD> ls

DATA/

FLASH/

OCR/

ASMCMD> cd FLASH

ASMCMD> ls

ZYZF/

ASMCMD> cd ZYZF

ASMCMD> ls

ARCHIVELOG/

BACKUPSET/

ASMCMD> cd ARC*

ASMCMD> ls

2014_06_29/
```

看来问题不在这里。

◆检查数据库实例

本地登录登录 oracle 客户端，说起来好专业，实际上就是执行 sqlplus 嘛。进行的操作记录如下：

```
[oracle@db50 ~]$   sqlplus / as sysdba


SQL*Plus: Release   11.2.0.1.0 Production on ??? 6? 29 12:01:47 2014


Copyright (c)   1982, 2009, Oracle.   All rights   reserved.




Connected to:

Oracle Database   11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production

With the   Partitioning, Real Application Clusters, Automatic Storage

Management, OLAP,

Data Mining and   Real Application Testing options


SQL> select   count(*) from v$process;


  COUNT(*)

----------

        41


SQL> select   count(*) from v$session;
```

```
  COUNT(*)

----------

        37
```

明明有连接嘛，看来问题不大。

◆检查 oracle 报警日志

在数据实例报警文件，发现如下有用信息：

```
<msg   time='2014-06-29T10:13:37.204+08:00' org_id='oracle'

comp_id='rdbms'

 client_id='' type='UNKNOWN' level='16'

 host_id='db50' host_addr='127.0.0.1'   module=''

 pid='5671'>

 <txt>************************************************************************

 </txt>

</msg>

<msg   time='2014-06-29T10:13:37.204+08:00' org_id='oracle'

comp_id='rdbms'

 client_id='' type='UNKNOWN' level='16'

 host_id='db50' host_addr='127.0.0.1'   module=''

 pid='5671'>

 <txt>Errors in

file   /u01/app/oracle/diag/rdbms/zyzf/zyzf2/trace/zyzf2_arc2_5671.trc:

ORA-19809: limit exceeded for recovery   files
```

```
ORA-19804: cannot reclaim 50331648 bytes   disk space from
42967498752 limit
 </txt>
</msg>
<msg   time='2014-06-29T10:13:37.204+08:00' org_id='oracle'
comp_id='rdbms'
 client_id='' type='UNKNOWN' level='16'
 host_id='db50' host_addr='127.0.0.1'   module=''
 pid='5671'>
 <txt>ARC2: Error 19809 Creating archive log file
to   &apos;+FLASH&apos;
 </txt>
</msg>
```

既然你说问题记录在文件

/u01/app/oracle/diag/rdbms/zyzf/zyzf2/trace/zyzf2_arc2_5671.trc，咱就乖乖听你的，打开文件看一下也无妨,打开一看，确实有用哟，其部分输出如下：

```
*** 2014-06-27   21:45:03.674 2046 krse.c
Archived Log   entry 770 added for thread 2 sequence 250 ID 0x1bcb54de
dest 1:   +FLASH/zyzf/archivelog/2014_06_27/thread_2_seq_250.443.
851377503
*** 2014-06-28   10:05:32.844 2046 krse.c
```

```
*** 2014-06-28  10:05:32.844
Archived Log  entry 782 added for thread 2 sequence 253 ID 0x1bcb54de
dest 1:  +FLASH/zyzf/archivelog/2014_06_28/thread_2_seq_253.546.
851421933
*** 2014-06-28  13:18:07.129 2046 krse.c


*** 2014-06-28  13:18:07.129
Archived Log  entry 795 added for thread 2 sequence 256 ID 0x1bcb54de
dest 1: +FLASH/zyzf/archivelog/2014_06_28/thread_2_seq_256.654.
851433487
*** 2014-06-28  14:41:21.362 2046 krse.c
```

原来是归档日志轮转的时候，没空间可以创建文件了。

◎故障处理

查明原因，因时间紧迫，需立即处理。过程记录如下：

◆检查闪回恢复区（为啥查它？因为归档日志在这里啊！）

```
SQL>  show parameter  db_recovery_file_dest


NAME                                 TYPE        VALUE
------------------------------------  ----------- ------------------------------
db_recovery_file_dest                string      +FLASH
db_recovery_file_dest_size           big integer   40977M
```

总共 40G，看一下实际用了多少？

```
SQL> select FILE_TYPE,PERCENT_SPACE_USED

from   v$flash_recovery_area_usage;


FILE_TYPE            PERCENT_SPACE_USED

-------------------- ------------------

CONTROL FILE                       0

REDO LOG                           0

ARCHIVED LOG                   97.75

BACKUP PIECE                   1.16

IMAGE COPY                         0

FLASHBACK LOG                      0

FOREIGN ARCHIVED LOG               0


7 rows selected.
```

◆删除归档日志

Rman target /

Delete archivelog all;

全部干掉。

再查看报警日志，开始有新的归档生成的记录：

```
<msg   time='2014-06-29T10:14:44.726+08:00' org_id='oracle'

comp_id='rdbms'

 client_id='' type='UNKNOWN' level='16'

 host_id='db50' host_addr='127.0.0.1'   module=''
```

```
pid='5538'>
 <txt>    Current log# 4 seq# 274 mem# 0: +DATA/zyzf/redo04.log
 </txt>
</msg>
<msg   time='2014-06-29T10:14:47.254+08:00' org_id='oracle'
comp_id='rdbms'
 client_id='' type='UNKNOWN' level='16'
 host_id='db50' host_addr='127.0.0.1'   module=''
 pid='5671'>
 <txt>Archived   Log entry 833 added for thread 2 sequence 273 ID
0x1bcb54de dest 1:
 </txt>
</msg>
```

从应用层面进行访问，一切都正常了。

补充：后边得把监控完善，再弄个计划任务，做好备份或者直接定期清理陈旧的归档日志。

架构设计分享之权限系统(看图说话)

作者：增洪亮 来源：http://knightswarrior.blog.51cto.com/1792698/1547232

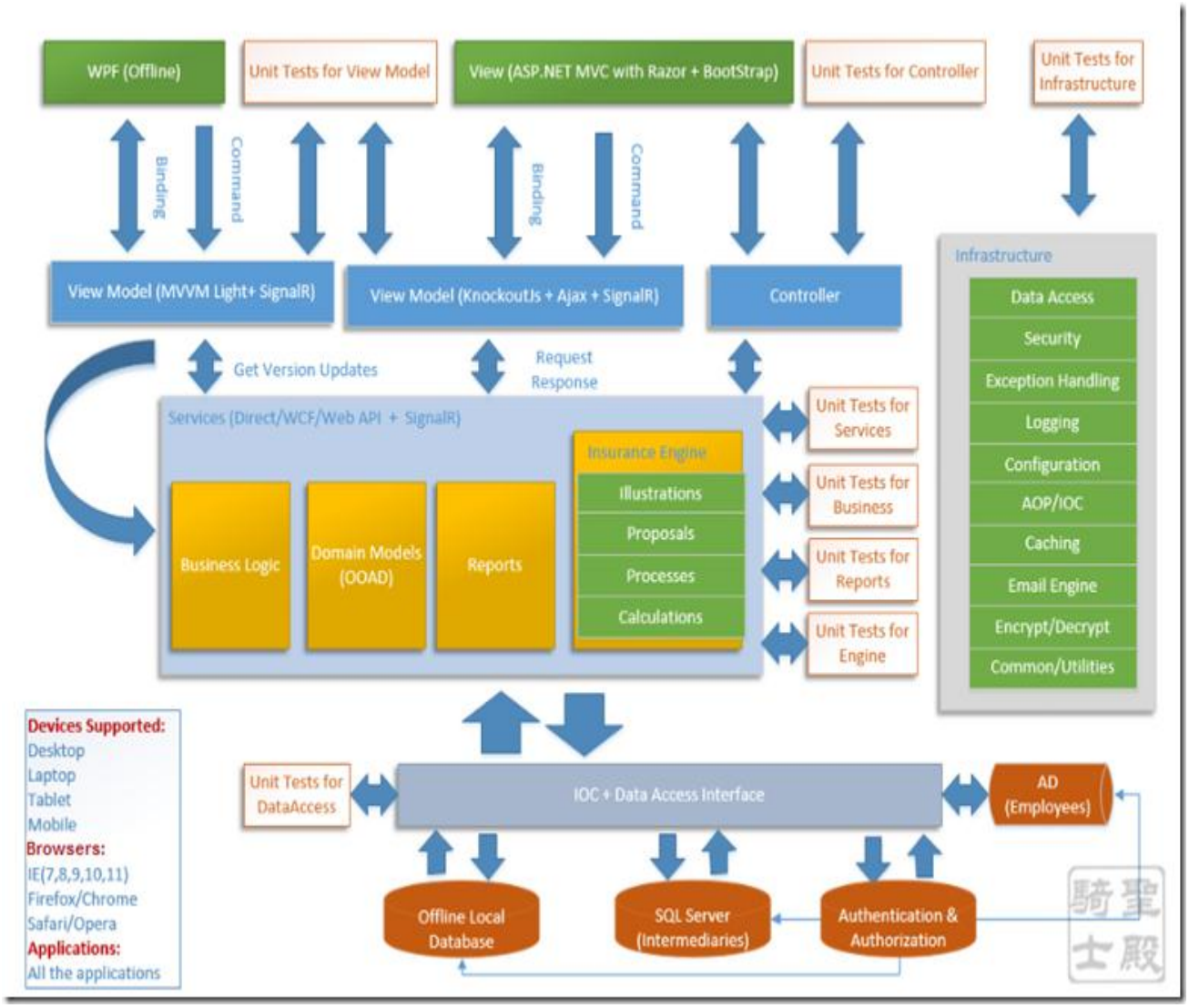
前面一篇文章

《最近架构随想》，我提到架构设计的一些构想，其实也是对之前项目经验的一些归纳及总结。今天我们就以权限系统作为切入点，谈一谈怎么设计权限系统以及怎么做到系统具有以下特性：

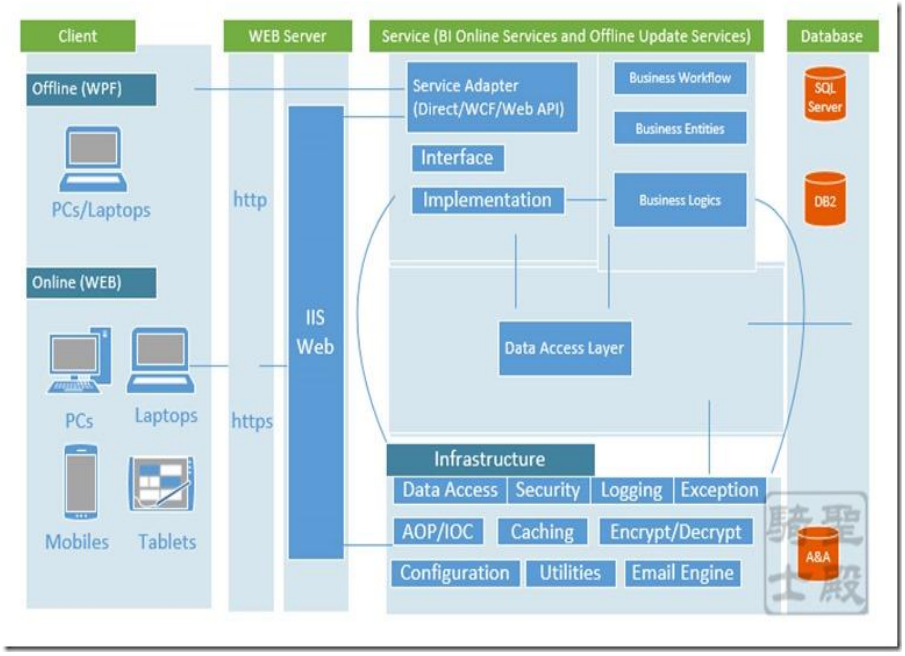
1. Organized：如果系统组织比较好，可以起到事半功倍的效果。
2. Encapsulated：对功能，结构，数据进行有效的封装，会使系统维护变得更加容易。
3. Reusable：对常用功能以及组件进行有效的封装，可以使系统变得结构清晰且方便维护。
4. Extensible：在设计系统的时候，如果很好的遵守 OO 的设计理念（OO 的五大原则 SOLID），即使系统做得很大，也会像火箭一样直冲云霄！
5. Replaceable：在很多时候我们需要考虑到系统，组件或者功能的可替换性，因为需求是会变的。
6. Testable：做到系统的可测性，会大大帮助开发以及维护，对团队开发以及分工协作起着非常重要的作用。
7. Loose Coupling：隔离耦合是架构设计必须要考虑的一个因素，如果系统不能做到高内聚、低耦合，那么在维护，升级，新功能开发方面就会是一场噩梦！
8. High Performance：高性能是系统设计必须重视的要点，用户不可能忍受简单页面加载超过十秒，也不可能接受页面操作频繁卡死的情形，所以在架构设计的时候必须从数据库，逻辑，服务以及 UI 进行合理的规划。
9. Scalability：如果能做到前面的几点，那么我有理由相信你的系统一定具备 Scalability。
10. Enjoy Your Life：最后一点也是最重要的一点，不要忙碌于重复的码农工作，喝杯咖啡，享受代码，早点回家，陪老婆、陪小孩，环球旅游，享受生活！

废话半天，下面就开始看图说话环节：

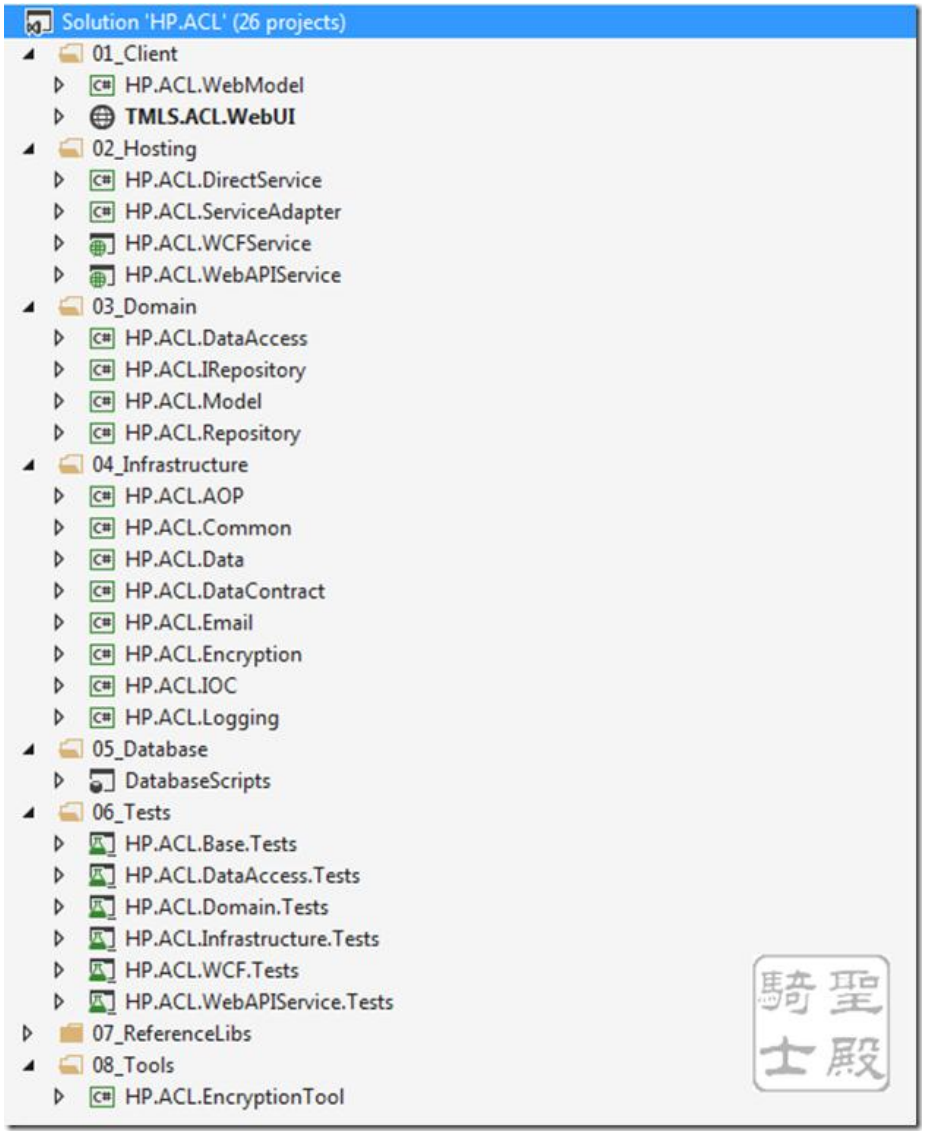
架构设计图：



部署及组件图：



详细解决方案：



01_Client：存放 UI 相关的项目，比如 Winform, WPF，ASP.NET， Silverlight，ASP.NET MVC 或者相关的 Web Model 及 View Model 项目。

02_Hosting：存放与 Service 相关的项目，可以是 Direct Service，Remoting Service，Web Service，WCF Service 或者 Web API Service 。

03_Domain：业务逻辑相关的所有实体以及操作（根据 OO 的思想设计类以及类之间的关系）。

04_Infrastructure：非业务方面的功能框架 （Data，Common，DataContract，AOP， IOC，Logging，Encryption，Email）。

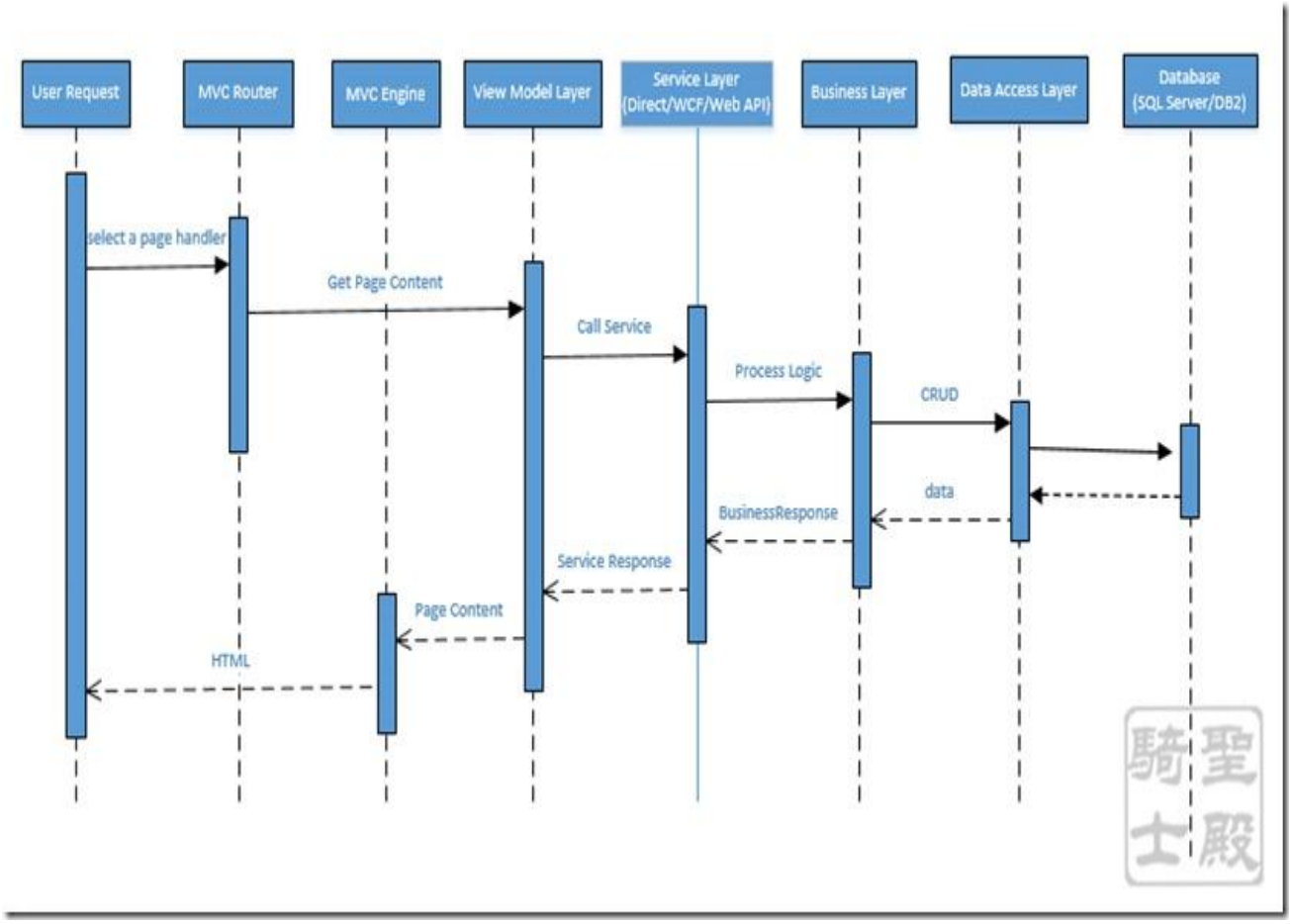
05_Database：数据库项目（包含所有数据库脚本，方便开发，部署以及维护）。

06_Tests：所有测试项目（数据访问测试，框架测试，业务逻辑测试，服务测试以及 View Model 测试）。

07_ReferenceLibs：项目相关的外部引用。

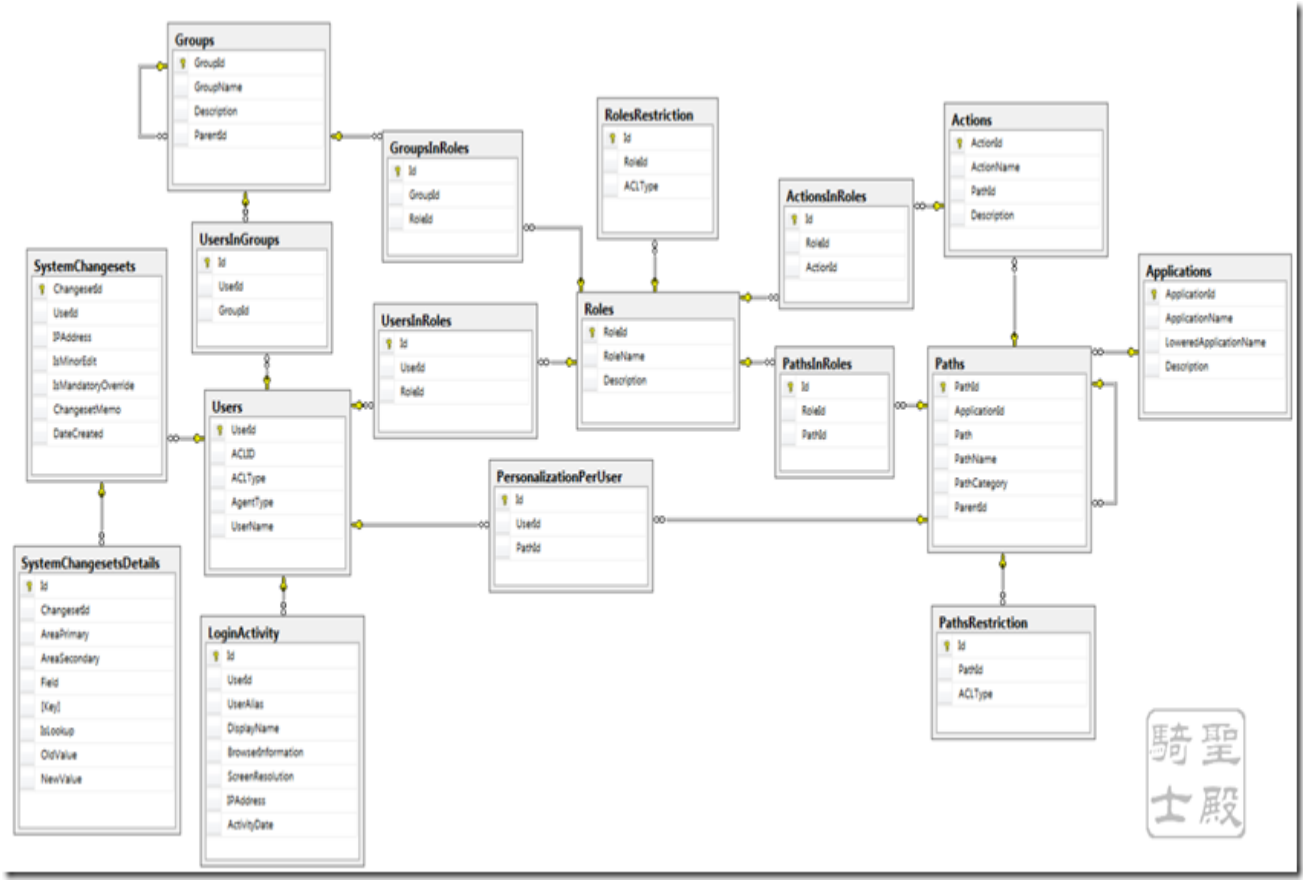
08_Tools：一些简单的工具，方便开发，测试以及部署。

各层执行序列（调用 Service 之前需要调用 Service Adapter，然后根据项目配置来决定调用 WCF Service，Web API Service 还是直接 DLL 引用）：

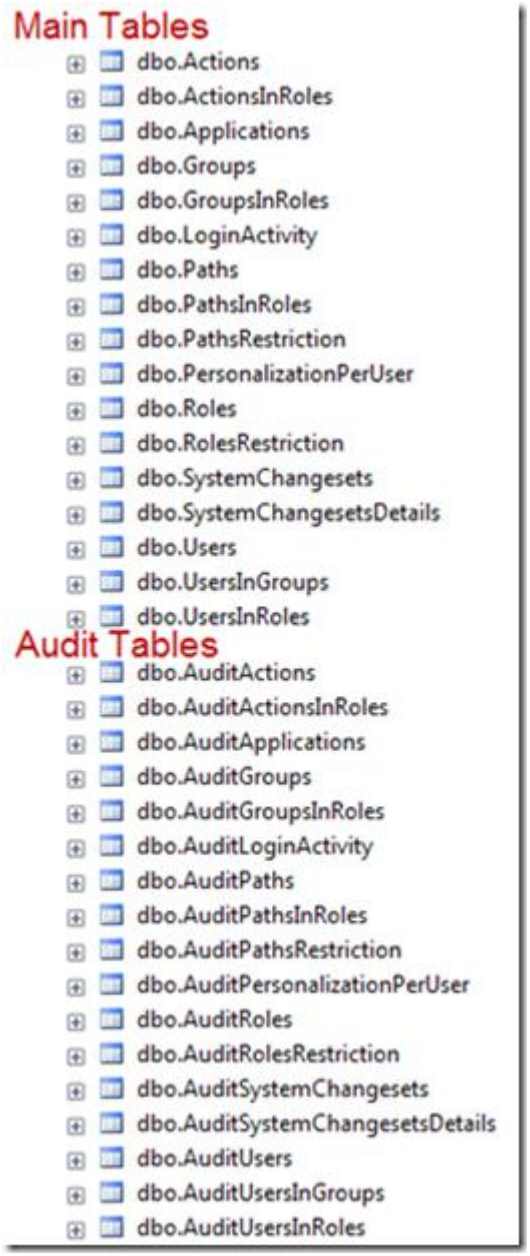


权限系统设计 （支持多系统，Module 多层级，Group 多层级，多用户来源，功能权限，数据权限等）：

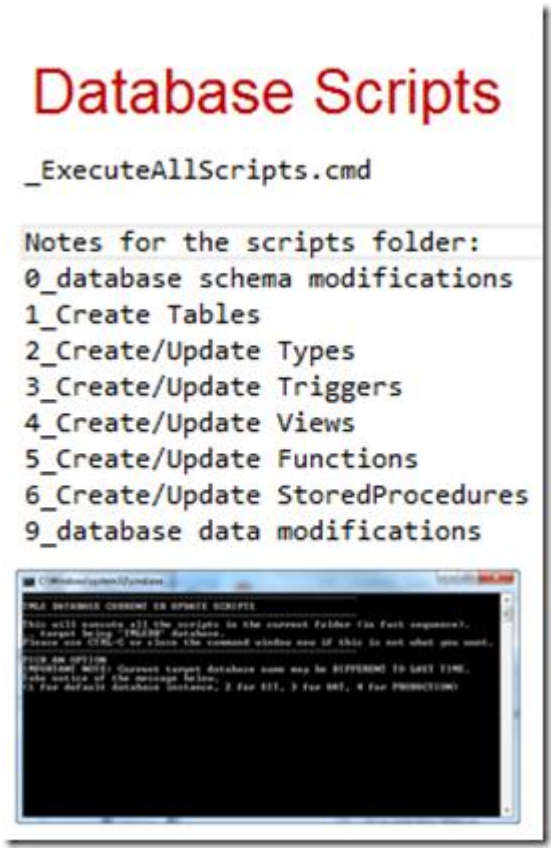
权限系统数据库关系图（三个中心点：Users, Roles, Paths）：



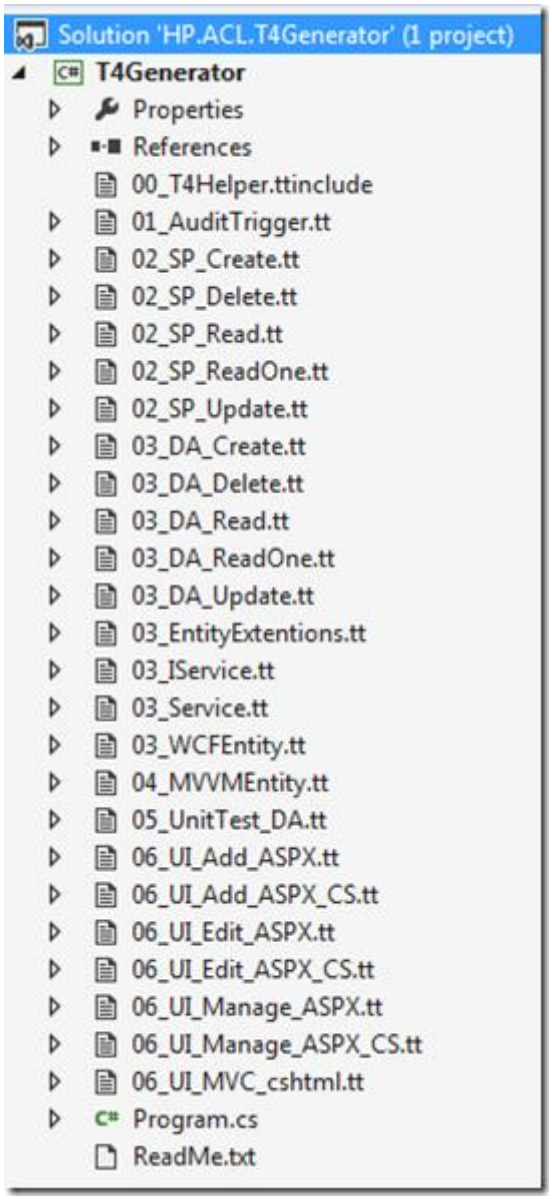
权限系统所有表（包括主要表以及备份表，主要是针对中小型项目，如果大型项目则要采用分库，分表以及分区的策略）：



数据库项目——脚本注意事项以及如何一键执行所有数据库脚本（方便管理数据库脚本并且对团队开发以及分工协作帮助很大）：



代码生成器——T4 完全生成数据库，业务实体，业务层，服务层以及 UI 代码 （之前也用 Winform 和 WPF 写过代码生成器，这是第一次使用 T4 生成整个项目，感觉非常好用，并且随时修改随时使用，而且还可以根据项目进行定制化）。



这篇文章主要是看图说话，所以如果你有什么反馈，心得或者建议，不妨留言，我会尽力在今天进行回复。

做技术到底可以做到哪种地步-技术为什么越走越

> 作者：汪洋　来源：http://yanyangtian.blog.51cto.com/2310974/1550495

尽管做技术已经有不少年头了，不管是犹犹豫豫还是坚定不移，我们走到了现在，依然走在技术这条路上。

不管我们处于何种职位，拿着哪种薪水，其实，我们会是不是的问问自己“做技术到底可以做到那种地步”，说的直白一点，其实我们很多人对技术这条路依然充满很多彷徨，不管我们的现状是多么的满意与辉煌。

最近一直招聘技术人员，见了很多求职的朋友，也和他们探讨了很多与职业发展，技术能力方面的问题，下面说下我个人的看法，和大家分享一下。

有很多的人总是一直在问“我搞.NET 很多年了，但是感觉现在越走越窄了”。

其实“越走越窄“主要可以从三个方面开看：

市场动向：从最近的招聘和很多的朋友的举动来看，特别是移动互联网的爆发和大数据技术，导致了很多的新人纷纷转型，而且很多做了多年技术的朋友，也跟着这股“趋势”在走，因为会认为“物以稀为贵”，会拿到不错的待遇。

我发现在我们大家都有一种“跟风”的习惯，包括我自己了。说到什么火爆，然后就大家一股脑的奔过去：很早以前，做网站很火爆，于是很多人都开始学习 HTML，随便说自己懂 HTML,都可以拿到不错的薪水；团购火爆，于是很多的创业者纷纷搞起团购，一时，“百团大战“；电商火爆，于是各地开始出现各种电商平台.....

一旦一片“蓝海”被搞成“红海”之后，市场就开始饱和，资源开始重新分配，其实最后依然是”82 理论“：20%的人占据着 80%的资源，依然会优胜劣汰，最后还是那句话”出来混的，早晚要还“。其实又说回来，最后选择，市场是一个很大的因素，另外，就是兴趣，真的是兴趣。或许，有朋友认为这点很扯，但是越到后面，这一点尤其重要，特别是当你的待遇，职位到了某个瓶颈之后，最后阻碍自己发展的就是自己。

思维定势：说到这一点，自己也算是深有感触。自己在做.NET 的时候，把自己的选择绑死在了 Windows 和微软技术上面，例如，为了搞负载均衡，不断的尝试 Windows 自带的 NLB，还是用 IIS 的 ARR（一直到 IIS7 才出来），在这些过程中出现很多问题，而且很多的情况，都无法满足，也想过买 F5 之类的产品。虽然自己在 Linux 环境下有很多成熟的软件和产品，如 HAProxy,LVS，但是一直对 Linux 有偏见，因为喜爱 Windows。

后面进入互联网公司之后，开始发现很多的情况选需要“混搭“，例如搜素采用 Solr，缓存采用 Redis，Memcached，日志采用 Kafaka，队列采用 RabbitMq。

后来要做用户分析和推荐方面，使用了 Hadoop+Mahout 等。

以前自己很天真：因为很多的开源组件都是 C 或者 Java 的，自己还尝试写一个.NET 版本的，最后发现能力有限，还是“拿来主义+代码定制“。

所以，要以开放的心态去做技术，这一点算是自己的很大的体会。

画地为牢：很多人总是会拿出所谓的技术人员“30“”35“岁等理论。也有很多人提到”转型“的问题，也有很多人提出“摆脱技术”的想法。

其实我们都很担心自己的技术生涯的长短，一般而言，就我自己的经历发现，不同的阶段培养自己不同的能力。

对于我们做技术的，不是说就每天呆呆的做技术，其实技术本身就是一个技能，通过做技术，锻炼出我们做事的方式和解决问题的能力。

举个例子，我们都经学生时代，我们永远记得我们学了很多的语文，而且老师每次讲课就搞什么场景分析，我记得在小学课本中有一个篇文章“小桔灯”，冰心老人鞋的，最后有这么一句“**我们都会好的**”。 这句话在我们看了，就是一句安慰人的话，但是老师在讲解的时候，就衍生了很多的含义：一方面告诉小女孩，她妈妈的病会好的，同时暗指了革命会胜利....

我们先不管老师的分析是多么的牵强，多么的让我们无语。后来，我们每次考试，总是有新一篇文章，让我们分析里面的很多“含义“。

其实，我们知道，学生时代的几十本语文，上千篇文章肯定不会全部出现在考试的“阅读理解“中，我们依然要学习千年不变的语文课本，其实就是在学习一种”分析的思维“，一种”举一反三“的能力。

唠叨了这么多，再说回来，我们学习技术，一方面是因为这些技术确确实实可以做出东西，这一点很不错，比我们学习的语文课本实用；另外一方面，也是在锻炼我们的思维，如何利用技术去解决问题，有个可以生搬硬套，但是很多需要变通。

例如，我之前在为很多公司做性能优化的时候，除了掌握必要的技术知识之外，另外就是思路：如何根据现状推断出问题所在，然后确定这个问题是否真实存在，然后收集数据分析，然后给出办法。

其实我们发现：技术能力是很重要，解决问题的思路同样重要。所以，以此类推，我们用这样的思维去破案，也可以：根据现场，找出线索，然后收集证据，然后抓嫌疑犯....

同样，医生看病也是这个思路.....

所以，很多的技术人员看中的是技术本身，没有跳出这个思维，最后看到的面就比较窄。

有朋友告诉我，他们在面试的时候分不同的层面，初级的人员，面试基本的编程知识；中级的就面试一些比较深的，偏架构，或者底层的知识；高级的，就看看他解决问题的思路，还有人品。

哟，一不小心，有唠叨了这么多，剩下的后续在讲吧。再次感谢大家！

技术变成客户才值钱

作者：刘源 来源：http://liuyuanljy.blog.51cto.com/607434/1548632

什么事与钱关联都显得有些俗，但没有钱又觉得这个世界这样的苦逼。作为一个技术人员，绝大多数人都在“苦逼”的生活中仰望“土豪”的生活，而唯一能够让我们达到这一目标的唯一途径就是将技术变成客户。

技术不值钱似乎成了一个不争的实事，大多数技术人员日思夜想着如何转型来获得更多的金钱，头发没有了，而故事依旧，为什么会出现这样的情形呢？

拿我现在来说，在企业混了多年，看清了企业中的是是非非，有一天我跟家人及朋友说，我不想干了，我要出来自己做，你们猜听到的人第一反应是什么？

无一例外，所有的人都在跟我说要慎重考虑，不要做出如此冲动的决定，在没有找到出路之前，还是先在原来的公司上班，这样才是最稳妥的做法。

我相信这是大多数人一直在做的方法，也是我原先采用的方法，别把自己逼上绝路。可是，如果真要出来创业，不把自己逼上绝路又有多少的成功几率呢？

我就是在这样的反复中到现在还没有实现自己创业的梦！

所以现在我绝不这样做了，我想把自己的技术变成客户，因为只有这样才能真正体现我的价值。

有人会说，在企业做现样的事情也能够体现价值，我赞同，但我不认同，因为这是两种不同的概念。

在企业做，体现的是被利用的价值，在外为自己做，才是真正体现自己本身的价值，因为这样才会让自己的能力和技术可以无限的放大。就像在鱼塘钓鱼，基本上知道会钓上什么样的鱼，而在江河中钓鱼，永远不知道下一条是什么样的鱼。

我非常感激家人、朋友、同事为我考虑这么多，但有几个人能够明白我真正想要做的是什么，金钱，不，金钱其实只是我想要的一种表达形式而已，我真正想要的远远超出金钱的范围。

我也知道从企业出来，没有了稳定的收入来源，生活会变得更加艰难，自己也会更加辛苦，但只有这样才能真正让技术变成客户，不然永远是一句空话。

实际上，在企业工作的过程中也做过些小项目，网络的、视频监控的、小型 PLC 自动化的、节能降耗的，甚至单个软件的销售等等，从这些小项目中所获得的成就感、技术以外的经验、金钱肯定要比工作中获得的多得多，只要在外做过项目的人应该都体会得到。其中一个最大的体会就是为什么有些人没有技术却能够在这一行中做得风声水起，根本的原因就是他开发或拥有了大量的客户资源。

一旦你拥有了客户资源，即使你不懂技术也可以获得良多，要是你懂技术，那么获得的会更加多。

在看这篇博文时，某些同志肯定会对号入座，会认为这样并不合适所有的技术人，我也是这样认为的。毕竟每个人的性格不同，性格决写思维，思维决定行为，行为决定命运！

如果你现在还在企业中工作，那么请相信那句“一切都是为了自己而工作，而不是为了老板而工作”。很多人不理解这句话，包括在我工作中我通常会说给我身边的人听，但没有做更多的解释。其实

道理很简单，在自己还没有掌握足够好或足够多的技术之前，利用企业这个平台努力积累，当积累到一定程度，你自然就会发现利用你所学所知完全可以做得更好和更多。

如果你和我一样一直在想如何才能做自己喜欢做的事，如果才能达到人生目标，那么在努力积聚力量的同时寻找实现自我的机会。

创业未必成功，但不创业肯定不会成功！

职场思考----大数据人才到底值钱在什么地方？

作者：bingyang87628 来源：http://bingyang.blog.51cto.com/533655/1544287

周末跟一做人才外包服务朋友聊天，提到自己正在学习大数据技术的时候。他直接就说到他现在有需求，但就是招不到合适的人才。然后提到说现在大数据人才的价值，收入，待遇方面。可以说基本上将近到 IT 行业的顶级了。不由得，就开始思考，大数据人才的价值到底在什么地方？

大数据思维

个人感觉，这是首先第一个需要有的。因为我们现阶段生活在一个数据爆炸的时代，掌握良好的数据思维是对你的商业决策，乃至 IT 架构有很大的帮助。比如说，我们现在的数据类型很多，数据量很大。但是我们用到的却很有限，而这些有限的数据又不能够让我们产生效益。所以，大数据思维很重要。我们对未来需求，乃至业务方向的理解都需要依靠大数据。这一点，并不一定是大数据技术，比如说，你企业累积的数据里只有十几 M 的 EXCEL 信息，我们也许不会什么线性回归，决策树，只用 EXCEL 里边的几个统计函数也许就能达到我们的数据分析目的。

还有一点，就是大数据思维可以帮助我们站在用户的角度去考虑问题。从而提升我们的销售率，转变传统的被动销售为主动销售。

再用到我们日常生活中，这种思维也可能帮助我们去学习最适合我们的技术。很多的技术学习都已经放在了网络上，这样就降低了我们学习的成本。但是资源虽然多，如何去伪存真？用有限的时间去学习更多的知识才是我们最应该掌握跟学习的。在这方面我设置的决策条件就是：由于技术的相通性，短期能够学会，能够在实际使用过程中用到。这项技术能够给自己创造相应的收入。关于看书，也有相应的决策条件：纸质书为主，技术类的纸质书一天 50 页左右。要有相应的思考，看完一本书要有相应的摘抄，有思考总结。尽量不要看电子纸，若看电子书，基本要求在 30 分钟之内可以看完的。

营销商业能力

实际上，一名真正的大数据人才，在技术上除了要出类拔萃外，在相应的商业模式上也要有一些自己的领悟与见底。说得简单点，就是销售的能力也要很好，尤其是在中国！不光要能讲出大数据的用途，方法，能为企业创造 的价值。而且还要能够很好地让大数据技术去落地，不要整天云里雾里，最重要的落地才是最为重要的。

这一方面，个人的理解就是：你技术再好不重要，只有适合我们企业的才是最重要的。也就是说在讲解技术的过程中相关的目的导向很重要。营销商业活动中最为重要的就是要成交，若你只是口若悬河的去讲技术实现，却不告诉对方你能够为对方创造的价值，这样无疑就是一次失败的讲解。

技术学习能力

说到技术，实际上大数据人才要掌握的技术是有一点难度，就从自身的角度来说，要掌握相应的编程能力最好。而现在的大数据平台主要基于的是 SPARK，部分早期的还有用到 HADOOP+MAPREDUCER 的。相应的数据挖掘主流是 SAS 与 SPSS,数据展现方面的是 R.而这一些软件的学习都要求我们有一定的编程基础。

其次要对架构有一定的掌握，比如说，数据仓库的架构，数据挖掘模型的架构，数据存储的架构，网络的架构，等等。

除了掌握这些，一些操作系统的底层内容。也就是一些硬件属性也要有一定了解。比如说存储的选择，操作系统的选择，优化，等等。

实际上，以上的任何一门技术都是可以达到高薪的程度，而大数据技术又是将这几项技术综合。所以，我们并不一定要全部掌握所有的技术，但要具备一专多能的能力，在用到新技术的时候，我们可以很快的学会。

实际上，基于以上三种能力，后期发展会有多方面的变化，比如说最基本的与人沟通能力，这一点也是做技术的人员可能最为稀缺的能力。总之，大数据人才也就是技术中上能力+营销高手的一个组合。其他的方面，还有外语能力，文档开发能力，讲课能力这一些，都可以通过后期的训练成就。

还有一点，就是现在大数据正火。而相应的，大数据人才能够在短期内为企业创造价值的能力也强调很高。

大数据人才值钱的最根本点，就是你能够为企业创造更大的收益。这种收益，可能是金钱上，可能是名誉上，也可能是活跃客户上。但是，一个大数据人才，你的学习，不能停止，将会随着业务需求的不同而不断改变。

扫描二维码，加 51CTO 博客为朋友

关注 51CTO 博客微信，打造你的个性！

http://blog.51cto.com

编后语

《51CTO 博客月刊》为 51CTO 博客整理出品

如果您有意投稿，请联系我们
如果您有反馈意见，请告诉我们

联系方式：
Email：blog@51cto.com
QQ: 1173854158

欢迎关注我们：
@51CTO 技术博客